Universidad Complutense de Madrid

Sistemas Informáticos

Curso 2008/2009

Aplicaciones del tratamiento inteligente de imágenes

Silvia Kalbakdij Sánchez

Pablo Lebrero Villar

Salvador Sánchez Rodríguez

Dirigido por:

Luís Garmendia Salvador

Departamento de Ingeniería del Software e Inteligencia Artificial

Facultad de Informática - Universidad Complutense de Madrid

Nosotros los autores del proyecto: *Aplicaciones del tratamiento inteligente de imágenes,* de la asignatura de *Sistemas Informáticos*:

Silvia Kalbakdij Sánchez DNI: 05324233-D

Pablo Lebrero Villar DNI: 32064479-H

Salvador Sánchez Rodríguez DNI: 50758743-C

Dirigidos por:

Luís Garmendia Salvador

Autorizamos a la Universidad Complutense de Madrid a utilizar y difundir, con fines académicos, el contenido de este documento de texto, así como del contenido del CD complementario que adjuntamos con él mismo.

Silvia Kalbakdij Sánchez

Pablo Lebrero Villar

Salvador Sánchez Rodríguez

Agradecimientos

A mis padres, por darme su apoyo incondicional y ayudarme a conseguir mis metas. A Fer por ser mi alegría y la energía que me anima todos los días. Por supuesto a todos mis amigos por siempre estar ahí y sobre todo por soportarme en los malos momentos.

Silvia.

A todos mis amigos que de una manera u otra me han dado alegría y ánimos para seguir adelante y han estado ahí siempre que los he necesitado. A mi familia por su ayuda y apoyo constante y todo aquel que ha influido de forma relevante a lo largo de mi vida. Gracias y siempre podréis contar conmigo.

Salvador

Quisiera expresar mi agradecimiento a quienes que de una forma u otra han hecho realidad este proyecto: a mis amigos por duplicar mis alegrías y dividir mis angustias, a mis profesores y compañeros por enseñarme que educación es lo que queda después de olvidar lo que se ha aprendido, a mi familia, especialmente a mis padres por darme su tiempo y paciencia día a día, y a Dios por darme todo lo que tengo que es más de lo que necesito.

Pablo.

Índice

Introducción

La obtención de información mediante técnicas de procesamiento de imágenes es un tema de permanente actualidad debido a que es posible aplicar a diferentes áreas de la ciencia y de la tecnología. Es por ello que la Visión Artificial se ha convertido en una de las áreas de trabajo de la Inteligencia Artificial con más futuro. Son muchos los campos de aplicación donde ya hoy se está utilizando con éxito, aportando un gran avance en los métodos de trabajo.

La visión artificial, no es más que un intento de aproximarse al funcionamiento biológico. En este caso, la luz reflejada en los objetos incide en la retina produciendo el plano de la imagen; esta imagen es proyectada sobre las terminaciones nerviosas fotosensibles, que llevan a cabo el procesamiento previo (visión preliminar), y posteriormente pasan la información al cerebro que realiza el procesamiento de alto nivel. En cuanto al funcionamiento artificial encontramos que la informática utiliza cámaras como sensores, de tal forma que éstas imitan el ojo humano, aunque no de forma tan sofisticada. Se basan en el principio de que la luz reflejada en los objetos pasa a través de una lente (iris) en un "plano de imagen" (retina) formando una imagen que puede ser procesada.

Utilizando técnicas de Visión Artificial se puede realizar un tratamiento *inteligente* de las imágenes recibidas en el ordenador en el que, más allá de la pura manipulación, se llega al análisis y reconocimientos de los objetos presentes en ellas. Esta visión artificial no es más que la adquisición automática de imágenes sin contacto y su análisis también automático con el fin de extraer la información necesaria para controlar un proceso o una actividad como: control de calidad, ordenación por calidades, manipulación de materiales, test y calibración de aparatos, monitorización de procesos, etc.

En la visión artificial, el objetivo es "comprender" la información obtenida por el dispositivo; aunque surgen problemas de enfoque en el plano si se usan lentes. Si se usan cámaras simples simplemente hay que invertir el plano. En definitiva el propósito es programar un computador para que "entienda" una escena o las características de una imagen.

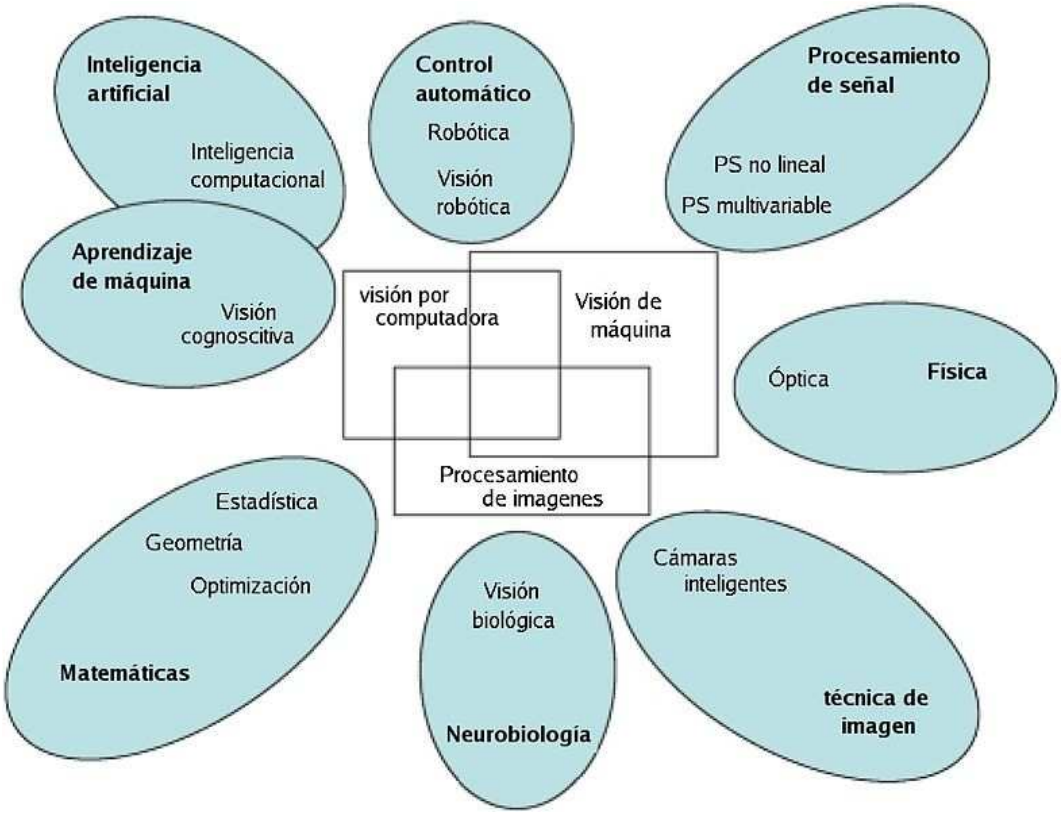


Figura 1: Esquema de relaciones entre visión por computadora y otras áreas afines.

1.1 Resumen

La Visión Artificial permite realizar un tratamiento *inteligente* de las imágenes recibidas en el ordenador de tal forma que se llegue al análisis y reconocimiento de los objetos presentes en ellas.

El proyecto muestra como a partir del tratamiento de las imágenes obtenidas en tiempo real, por una cámara, se capta el movimiento de un objeto seleccionado para posteriormente llevar a cabo tres aplicaciones. La primera de ellas consiste en representar la evolución o trayectoria del objeto captado, de manera gráfica. La segunda se refiere a la simulación del entorno del escritorio de un ordenador donde a través de la cámara de selecciona una carpeta y se desplaza a otro punto del escritorio. Por último la tercera aplicación consiste en el teclado de un teléfono, donde se puede marcar un número, llamar y colgar; todas estas acciones de manera simulada sobre un formulario gráfico.

***Palabras claves*:**

Imagen; Luz; Color rojo; WebCam; Movimiento; Trayectoria; Selección; Detección.

1.1 Summary

The Artificial Vision allows realizing an intelligent treatment of the images received in a computer in order to analyze and to recognize present objects in them.

This project shows how from the treatment of the images obtained in real time, by a camera, The movement of an object selected can be perceived in order to obtain the implementation of three applications. The first one of them consists of representing the evolution or path of the caught object, of a graphical way. The second one refers to the simulation of the environment of a computer desktop where you can select a folder and move it to another point of the computer desktop thanks to the camera. Finally the third application consists of the keyboard of a telephone, where it is possible to mark a number, call and hang up; all these actions are simulated in a graphical form.

Keywoards:

Image; Light; Red color; WebCam; Movement; Path; Selection; Detection.

1.2 Idea inicial

El objetivo inicial fundamentalmente consiste en la captación del movimiento a través de una WebCam para luego simular e interpretar el funcionamiento de un ratón sobre la pantalla de un ordenador.

A pesar del desarrollo tecnológico que estamos viviendo a nivel mundial, la extensión del uso de los ordenadores en prácticamente todos los ámbitos de trabajo y a nivel de usuario, aún hay gente que no sabe utilizar uno sin que le parezca complicado.

Con este proyecto se busca ayudar y promover la facilidad de la utilización de dispositivos que actualmente puede encontrarse cualquiera en muchos momentos de su vida.

Intentamos conseguir que cualquier usuario pueda manejarse con las herramientas y múltiples elementos que puedes encontrar en el escritorio de su ordenador de forma tan sencilla que parezca que son sus propias manos las que están manipulando todos estos elementos.

El propio usuario podría por ejemplo mover, maximizar, priorizar ventanas con el simple hecho de mover su mano alrededor de la pantalla. Entonces, aprendería con mayor rapidez a comprender y ejecutar la gran variedad de acciones que puede realizar debido a que es él mismo el que está usando su propio cuerpo y movimientos para manejarse con el entorno virtual. La eficacia este aprendizaje radica en que el usuario simularía acciones que realiza diariamente con objetos que tiene en su entorno real como escribir con un lápiz o desplazar a éstos conforme a su agrado.

En definitiva, intentamos familiarizar a todos los usuarios con el "uso del ratón del ordenador" y aportarles una gran ayuda para introducirse en la era de las nuevas tecnologías de una forma más sencilla y cómoda, libre del uso del ratón.

También hay que resaltar que para el uso de la aplicación sólo son necesarios unos requisitos mínimos al alcance de cualquier usuario interesado; con un simple dispositivo de captación de imágenes (WebCam) y un ordenador con un procesador estándar para el avance que hay en nuestros días, cualquier persona puede tener una herramienta tan comprensible como es la aplicación.

1.3 Motivaciones

Nuestra motivación e interés tiene como origen el interés por la Realidad Virtual. La realidad virtual (RV) es un tipo de tecnología que puede ser definida como un espacio tridimensional gráfico, táctil, visual o auditivo (también llamado ambiente o "mundo" virtual) generado por computadora, donde los usuarios pueden interactuar y navegar en ese espacio utilizando aparatos especiales. La RV presenta tres características esenciales: Ésta produce el efecto de inmersión; esto es, cuando el usuario siente que está "adentro" o presente en el ambiente virtual, es interactiva, porque podemos seleccionar o manipular algún objeto dentro del ambiente virtual y es multisensorial, cuando los usuarios del ambiente virtual usan más de un sentido sensorial para interactuar con éste.

Además de poder utilizarse por una persona, un ambiente de realidad virtual puede ser accedido por dos o más personas. La realidad virtual de colaboración (también llamada multiusuario, o con ambientes virtuales distribuidos) es un sistema de RV que contiene un ambiente virtual gráfico compartido entre varios usuarios, pudiendo funcionar en una red local o amplia (Internet), en el cual los usuarios se pueden comunicar por medio de ventanas de mensajes de texto (llamado *chat* en inglés), gestos, o voz síncrona y dichas maneras de comunicarse generalmente se encuentran embebidas o son parte del sistema de realidad virtual. Este tipo de realidad virtual también podemos llamarla *concurrente*, ya que puede ser usada de manera cooperativa, donde dos o más personas pueden interactuar en conjunto hacia una meta u objetivo en común. En el contexto de la tecnología de redes de computadora, un ambiente de realidad virtual inalámbrico es una aplicación móvil de un ambiente virtual de colaboración, la cual utiliza una red inalámbrica para el intercambio de datos entre los participantes del ambiente virtual, incluyendo el envío de la voz sobre IP (VoIP, por sus siglas en inglés), la distribución de imágenes, gestos, sonido, texto y gráficas dentro del ambiente virtual. La red inalámbrica puede ser accedida por medio de computadoras portátiles y en un futuro próximo inclusive por medio de asistentes personales digitales (PDA).

El concepto de realidad virtual no es nuevo. Fue creado en los años cincuenta y cristalizado en los años sesenta cuando Morton Heilig, un cineasta estadounidense, concibió, describió y patentó por primera vez un aparato al que le llamó Sensorama, parecido a un video juego. Éste constaba de una proyección de una película que podía ser vista por una persona, percibiendo aromas, viento y movimiento dentro del mismo, como parte de la recreación de la película proyectada dentro del aparato (Sherman y Craig, 2003). La realidad virtual no tuvo avances significativos sino hasta principios de los años ochenta, debido a que hasta esas fechas las computadoras personales no eran lo suficientemente poderosas en cuanto a capacidad de procesamiento, memoria y despliegue gráfico. Actualmente, la realidad virtual se aplica en muchos campos del conocimiento y la técnica, desde la industria química y farmacéutica, pasando por la medicina y la psicología, hasta la exploración de petróleo.

La ventaja principal de la RV de colaboración, y en especial si el acceso es móvil por medio de un ordenador portátil, es que se puede usar a cualquier hora, en cualquier lugar. Las tendencias de su uso a futuro indican que se aplicará con mayor frecuencia en el entrenamiento técnico y en la educación en general con una mayor integración de información sensorial en computadoras móviles con ambientes virtuales, utilizando eficientemente el ancho de banda en redes inalámbricas.

Parte 2. DESARROLLO

2.1 Conceptos Teóricos

Manejo de Imágenes

Las imágenes obtenidas por medios artificiales utilizan como estructura básica para su representación, una matriz de índices que hacen referencia a los elementos de una paleta. A su vez, la paleta es otra matriz, en la que cada columna representa el valor para un componente de color y cada fila es un color utilizado por la imagen.

El plano de la imagen se suele dividir en partes iguales (píxeles) típicamente podemos considerarlos de forma rectangular, una cámara típica obtiene imágenes de 512x512 píxeles mientras que en una retina hay $120x10_6x6x10_6$ terminaciones. El valor de cada píxel es proporcional a cantidad de luz reflejada por la parte de la superficie del objeto que se proyecta sobre ese píxel y a su vez depende de la reflexión especular (reflejada directamente) y la reflexión difusa (absorbida y re-emitida).

Las características de una imagen están determinadas principalmente por: el material del objeto, la posición de las luces en la escena y reflejo de otros objetos en la escena.

También se puede destacar que toda imagen cuenta con una serie de formatos, en función del color, la iluminación y la compresión de la imagen o la información de la misma.

Por un lado se encuentra el formato de iluminación, que maneja el tono, la saturación y la intensidad de la luz en la imagen; este formato puede ser HSB (HSI, HLS, HSV, etc.) Hue (tono), Saturation (saturación) y Bright (intensidad).

Por otra parte se encuentran los formatos de compresión con o sin pérdida de información, como por ejemplo JPG. Y que en este caso es el utilizado para las imágenes del proyecto, debido a que se considera que el formato **jpg** es mejor para fotografía digital mientras que los formatos .gif y .png son mejor para imágenes gráficas.

Por último en cuanto al formato de color encontramos el RGB; es un formato aditivo que mezcla cromaticidad e iluminación.

La descripción RGB (del inglés *Red, Green, Blue*; "rojo, verde, azul") de un color hace referencia a la composición del color en términos de la intensidad de los colores primarios con que se forma: el rojo, el verde y el azul. Es un modelo de color basado en la síntesis aditiva, con el que es posible representar un color mediante la mezcla por adición de los tres colores luz primarios.

Para indicar con qué proporción mezclamos cada color, se asigna un valor a cada uno de los colores primarios, de manera, por ejemplo, que el valor 0 significa que no interviene en la mezcla y, a medida que ese valor aumenta, se entiende que aporta más intensidad a la mezcla. Aunque el intervalo de valores podría ser cualquiera (valores reales entre 0 y 1, valores enteros entre 0 y 37, etc.), es frecuente que cada color primario se codifique con un byte. Así, de manera usual, la intensidad de cada una de las componentes se mide según una escala que va del 0 al 255.Por lo tanto, el rojo se obtiene con (255,0,0), el verde con (0,255,0) y el azul con (0,0,255), obteniendo, en cada caso un color resultante monocromático. La ausencia de color —lo que nosotros conocemos como color negro— se obtiene cuando las tres componentes son 0, (0,0,0).

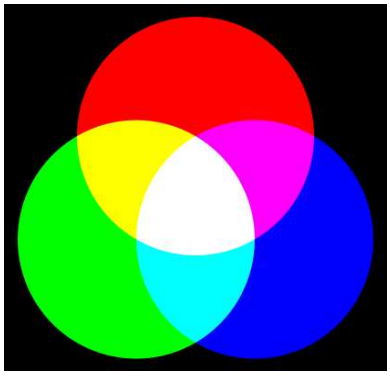


Figura 2: Mezcla aditiva de colores

El Tratamiento de imágenes a alto nivel trata de distinguir “cosas” a través de una serie de pasos. Consiste en una fase inicial o preprocesado, seguida del análisis de la imagen o segmentación de la misma, su representación, descripción y finalmente clasificación.

La fase del *preprocesado* está constituida por un conjunto de métodos que tienen la finalidad de realzar en la forma aquellas características que nos interesan para una aplicación concreta. La idea se basa en desechar todas aquellas características que no sean útiles para el proceso al que pretendamos someter a la imagen, al tiempo que se deben conservar aquellas que en alguna medida se consideren importantes.

La *segmentación* es una tarea de fundamental importancia en el análisis de imágenes. Su objetivo es dividir la imagen en partes, cada una de las cuales se corresponde con un objeto de la realidad. En general, los métodos de segmentación se basan en el hecho de que los píxeles que forman un objeto mantienen una cierta similitud entre ellos y, sin embargo, en las fronteras que separan dos objetos aparecen mayores variaciones.

La *representación* de una región en una imagen comprende dos tipos de procedimientos:

- Representación mediante las características externas (contorno).
- Representación mediante las características internas (los píxeles que comprenden la región).

El seleccionar un esquema u otro de representación vendrá determinado por el área de aplicación, así como, el tipo de problema que se desea resolver.

El problema de la *clasificación* consiste básicamente en lograr una partición del espacio de características en regiones mutuamente excluyentes y en asignar a cada región una clase.

Figura 3: Esquema del proceso seguido en la construcción del proyecto

Librería utilizada

La librería utilizada es Java Media Framework (JMF), que es una extensión de Java para poder realizar tareas multimedia con más facilidad en este lenguaje de programación. Esta especializada en las tareas multimedia que requieren un tiempo corto de respuesta o incluso requieren esta respuesta en tiempo real.

Para este proyecto se ha utilizado la versión JMF 2 que permite capturar y almacenar datos multimedia y controlar el tipo de procesamiento durante la retransmisión entre otros. Los objetivos del API JMF 2 con respecto a la JMF 1.0 que nos hacen decantarnos por esta versión son:

- Programar con facilidad.
- Permitir la captura de datos multimedia.
- Permitir el flujo multimedia y las aplicaciones con conferencia en Java.
- Permitir a los desarrolladores avanzados y proveedores implementar soluciones personalizadas basándose en los API existentes e integrar con facilidad características nuevas con el código actual.
- Proveer el acceso a los datos multimedia “raw” (Son datos multimedia “en crudo” sin utilizar ningún tipo de artefacto para tener que tratarlos)

- Permitir el desarrollo de codecs, efectos de procesamiento, multiplexores, demultiplexores y "renderers" personalizados.
- Mantener la compatibilidad con JMF 1.0.

Por otra parte, es necesario instalar JMF como un paquete adicional ya que no se incluye en la JDK o JRE de la maquina virtual de java.

Características

Sus características principales son:

- Gran estabilidad, al estar funcionando sobre la máquina virtual de java.
- Sencillez, ya que permite, usando unos pocos comandos, realizar complejas tareas multimedia.
- Potencia, permitiendo la manipulación de elementos multimedia de vídeo y audio locales (procedentes de la misma máquina en la que se ejecuta el programa), así como la retransmisión en tiempo real de vídeo y audio a través de la red mediante el protocolo RTP.

Licencia

El código de JMF esta accesible bajo el programa Sun Community Licensing (SCSL), que puede resumirse según sea o no para uso comercial.

2.2 Aplicaciones

Se han realizado tres aplicaciones.

La primera aplicación está basada en la detección de la trayectoria de un objeto marcado: en el formulario se va dibujando el trazo marcado por el usuario utilizando el objeto en cuestión para posteriormente poder guardar la imagen de la trayectoria en un archivo .JPG con la intención de que en un futuro esta sirva para ser reconocida por un programa OCR.

La segunda aplicación simula el escritorio de un ordenador. En ella, se puede realizar un clic con el objeto marcado sobre el formulario para realizar el desplazamiento de una carpeta a través del escritorio tal y como se haría con el ratón.

La última aplicación está basada en un uso práctico y útil del uso del clic de tal forma que se simula el uso de un teléfono convencional. Se pueden marcar los números para llamar a un número de otro usuario y posteriormente realizar las acciones de llamar y colgar similar a como se haría con un teléfono doméstico.

2.2.1 Detección de la trayectoria del objeto marcado.

Breve explicación

El entorno en el que se desarrolla la aplicación es la propia pantalla del ordenador.

Los usuarios pueden verse en una pantalla que muestra el entorno real en el que se encuentra la persona captada por la WebCam. En dicha pantalla, se podrán visualizar todos los movimientos realizados por el usuario, y determinados por llevar una marca de color rojo, de manera que se puedan distinguir, del mismo modo el usuario podrá guiarse a través de su propia imagen para así seguir la trayectoria deseada.

El entorno actúa en forma de espejo para facilitar el uso del mismo, es decir, si el usuario mueve su mano derecha es que quiere alcanzar algún objetivo que se encuentre en la parte derecha de la pantalla y viceversa.

Todas las coordenadas de la ventana en la observamos nuestros movimientos tienen su correspondencia matemática con respecto a la imagen captada por la WebCam de tal forma que las acciones que realicemos corresponden de la forma más correcta posible con el lugar en el que el usuario quiere realizar dicha acción.

La aplicación se basa en seguir el movimiento de un objeto con un determinado patrón, en este caso de color rojo, de manera que al tratar la imagen se pueda distinguir la trayectoria que sigue el objeto, y de esta forma representarla gráficamente, a través de un formulario. Este formulario gráfico muestra la trayectoria que realiza el objeto en la secuencia de imágenes capturadas del video en ejecución, de manera que se visualiza una línea en función del movimiento del objeto.

En definitiva lo que se muestra en la interfaz gráfica de la aplicación, consiste en dos ventanas, una que es el video captado por la WebCam y otra ventana que es el formulario donde se va trazando la trayectoria que sigue el objeto rojo tratado por las imágenes tomadas de ese video.

Desarrollo

Antes de comenzar con el desarrollo de esta fase, se acordaron los puntos clave a tener en cuenta.

El primer punto a tratar fue escoger el color del objeto para el cual desarrollamos nuestra aplicación. Decidimos escoger un color muy bien diferenciado del resto de la gama de colores para facilitar su detección. La elección fue uno de los tres colores integrantes de la codificación del color de los píxeles RGB: *el rojo.*

Antes de que se muestre la imagen capturada por la WebCam en cada momento, tenemos un menú de opciones para poder configurar las características de la captura. Decidimos escoger un Frame Rate adecuado y un tamaño del video para poder optimizar el procesamiento de la imagen de nuestra aplicación.

Para el estudio de la trayectoria del objeto guardamos las imágenes capturadas por la WebCam cada un cierto tiempo comprendido entre 0,5 y 2 segundos y posteriormente tratamos cada una de ellas.

Para tratar cada imagen, filtramos todos aquellos objetos que no se corresponden con el objeto a tratar.

Para ello, se calculó el área de cada uno de ellos y se comparó con los valores que caracterizan a aquel que nos interesa. Posteriormente se determinó el punto central de aquel objeto que nos es necesario para su representación gráfica.

A parte da la pantalla que corresponde a la captura de la cámara Web, se ha creado otro formulario para representar adecuadamente la trayectoria del objeto marcado. Para conseguirlo, se guarda en cada momento el centro de dicho objeto y se une con el centro que tenía en la anterior captura y así podemos determinar en cada momento donde se encuentra y de desde donde provenía anteriormente.

Partes de la aplicación

Con el propósito de detectar la trayectoria del objeto marcado, se incluye en el proyecto las siguientes clases:

La **clase Píxel** para determinar el color de cada punto que formaba la imagen capturada a través de nuestra cámara Web. Sus atributos propios son las componentes rojo, verde y azul correspondientes al RGB. Su importancia radica en la selección de aquellos píxeles correspondientes al color que hemos determinado para el patrón.

Además la **clase Punto** se implementa para determinar las coordenadas X e Y de los puntos que caracterizan múltiples atributos de las clases del proyecto. Esta clase cuenta con una función que optimiza el tiempo de ejecución pues agrupa puntos de la imagen pertenecientes al mismo grupo para poder diferenciarlos de otros que no pertenecieran al mismo y con ello poder distinguir los objetos que se encuentran en la imagen en cada momento.

Para guardar las características de todos aquellos objetos candidatos que aparecen en las capturas, se incluye una **clase Objeto**, es decir, para poder guardar aquellos que tienen el mismo color que el patrón. Incluye funciones para calcular el área correspondiente a cada uno de ellos y para poder determinar la adyacencia de sus puntos. La relevancia de esta clase radica en la determinación de todos los objetos candidatos a ser el que se quiere estudiar y el posterior filtrado de todos aquellos que no cumplan las condiciones.

También la **clase Imagen** es utilizada para tratar las capturas que se hacen a través de la WebCam y con ella poder determinar el centro correspondiente del objeto rojo a distinguir. Esta clase tiene los siguientes atributos:

- **Matriz de Píxeles**
  Se utiliza esta matriz para guardar los Píxeles que forman la imagen a tratar en cada captura para su posterior tratamiento.

- **Centro**
  Corresponde al centro del objeto del cual se quiere seguir su trayectoria.

- **Esquinas del objeto principal**
  Se utilizan para poder filtrar todos aquellos objetos que tengan el color del patrón pero no correspondan al tamaño adecuado de aquel que se quiere estudiar.

- **Objeto principal**
  Pertenece a la clase objeto y corresponde a aquel que es útil para la aplicación.

- **Lista de Objetos**
  En ella se guardan todos aquellos elementos de la escena que contienen el color escogido para el patrón. A partir de ella se puede hacer el filtrado comentado con anterioridad.

En esta clase se especifican las restricciones en cuanto a la selección del Píxel respecto a su color. Cada componente RGB abarca un rango de 0 a 255 en números enteros pues la representación de cada una de ellas en código binario ocupa exactamente 8 bits (2^8 =256); para la restricción mencionada con anterioridad hemos determinado seleccionar aquellos píxeles que tengan una componente roja mayor a 200, una componente verde menor a 60 e igualmente para la componente azul.

Para la selección de cada componente de la codificación RGB de los píxeles almacenamos cada imagen dentro de un buffer para posteriormente escoger los bits que corresponden a cada una de ellas; es decir, los primeros 8 bits corresponden al rojo, los siguientes a la componente verde y desde el bit 16 al 23 determinan el grado de azul que existe en ese Píxel.

Esta clase también cuenta con la función más relevante para la selección del objeto de estudio: *cuentaRojo*. Esta función se encarga de encontrar aquellas regiones de la escena que abarcan un gran número de píxeles de color Rojo (correspondientes a las restricciones) y guardarlas con sus correspondientes atributos en la lista de Objetos. Después se comparan todos los objetos de la lista entre sí y sólo se selecciona aquel que corresponde con las características que se han establecido para que sea único.

La **clase JMStudio2** es la más importante de la aplicación. En ella se encuentran las múltiples funciones y variables que permiten la visualización de imágenes provenientes de las capturas de la cámara Web y sus funciones más relevantes se establecen como la raíz de toda la estructura de llamadas a funciones que se realizan para la ejecución de la aplicación.

Sus dos funciones mas significativas son *captureMedia () y doSnapShot ()*. Funciones que utilizamos gracias a la ayuda de la librería JMF.

> **captureMedia:** Con esta función que nos aporta la librería JMF capturamos información de dispositivos como micrófonos o cámaras Web, como es el caso. Los datos obtenidos pueden ser procesados y utilizados o almacenados para tratarlos en otro momento.
>
> A través del *CaptureDeviceManager* se localiza el dispositivo (cámara Web) de la que se obtiene la información *CaptureDeviceInfo* que se utiliza para crear la *DataSource* y posteriormente se crea el procesador que utiliza esta *DataSouce* y se inicia la captura llamando a la función *doSnapShot ( ).*
>
> Dentro de esta función también se define un *timer* que determina la velocidad de ejecución de la cámara, de la misma forma que se crea el *JFrame* o ventana donde se muestra el video capturado en cada momento por la cámara Web.

La importancia de este método radica en la determinación del punto central del objeto marcado y el trazado de todas las líneas que describen la trayectoria de dicho objeto.
Extracto de código de importancia relevante:

```
timer= new Timer(100,new ActionListener(){

            public void actionPerformed(ActionEvent e){

                  Punto centro = null;

                  doSnapShot();

                  if(image != null){

                        if(imagenCamara == null)
                              imagenCamara = new Imagen();

                        imagenCamara.setImage(image);

                        int a=imagenCamara.cuentaRojo();

relacionVentanaImagenX=
(double)panelDibujo.getWidth()/(double)imagenCamara.getWidth();

relacionVentanaImagenY=
(double)(panelDibujo.getHeight()/1.2)/(double)imagenCamara.getHeight();

                        JPanel panel = (JPanel)ventana.getContentPane();
                        panel.setLayout(null);
                        ventana.setVisible(true);
                        //Recogemos el centro de la figura
                        centro = imagenCamara.getCentro();
                        if (centro!=null){
                              if (centroAnteriorImagen==null){
                                    centroAnteriorImagen=new int [2];

                                    centroAnteriorImagen[0]=centro.getX();
                                    centroAnteriorImagen[1]=centro.getY();
        //Incluye en el arrayList el primer centro aunque aun no se ha
dibujado nada Escrito.add(
new Punto((int)(panelDibujo.getWidth() -
centro.getX()*relacionVentanaImagenX),
            (int)(centro.getY()*relacionVentanaImagenY)));
                              }
                              else{

                                    BufferedImage dibujo = new

      BufferedImage(panelDibujo.getWidth(),panelDibujo.getHeight(),
                              java.awt.image.BufferedImage.TYPE_INT_RGB);
                              //Graphics g = dibujo.getGraphics();
                                    if(grafico==null)
                                          grafico = dibujo.getGraphics();



                                    panelDibujo.setBounds(0,
0,ventana.getWidth(),ventana.getHeight());
                                    int oldX=(int)(panelDibujo.getWidth()
- centroAnteriorImagen[0]*relacionVentanaImagenX);
                                    int
oldY=(int)(centroAnteriorImagen[1]*relacionVentanaImagenY);
```

```
                                    int newX=(int)(panelDibujo.getWidth()
- centro.getX()*relacionVentanaImagenX);
                                    int
newY=(int)(centro.getY()*relacionVentanaImagenY);

                                    //Añadimos el nuevo centro de la
imagen trazada
Escrito.add(new Punto(newX,newY));


      panelDibujo.getGraphics().drawLine(oldX,oldY,newX,newY);

                                    centroAnteriorImagen[0]=centro.getX();
                                    centroAnteriorImagen[1]=centro.getY();
                              }

                        }
                  }
            }
        });
        timer.start();
        }
```

doSnapShot: Con ella se consigue la captura de las imágenes. En primer lugar, se crea un buffer para capturar la imagen que se muestra por pantalla en un determinado momento; este buffer se transforma en otro buffer característico para tratar con imágenes y posteriormente guardarlo en la carpeta raíz en la que se encuentra el proyecto. El archivo de salida tiene como nombre “salida.jpg”. La elección del formato de dicho archivo se establece para tratar con imágenes con suficiente calidad pero que no abarque mucho espacio para poder ser procesada lo más rápido posible.

UML

A continuación modelamos la aplicación a través de diagramas de clases, que incluyen información sobre la relación entre los objetos de la aplicación, las propiedades de los mismos y su conjunto de operaciones.

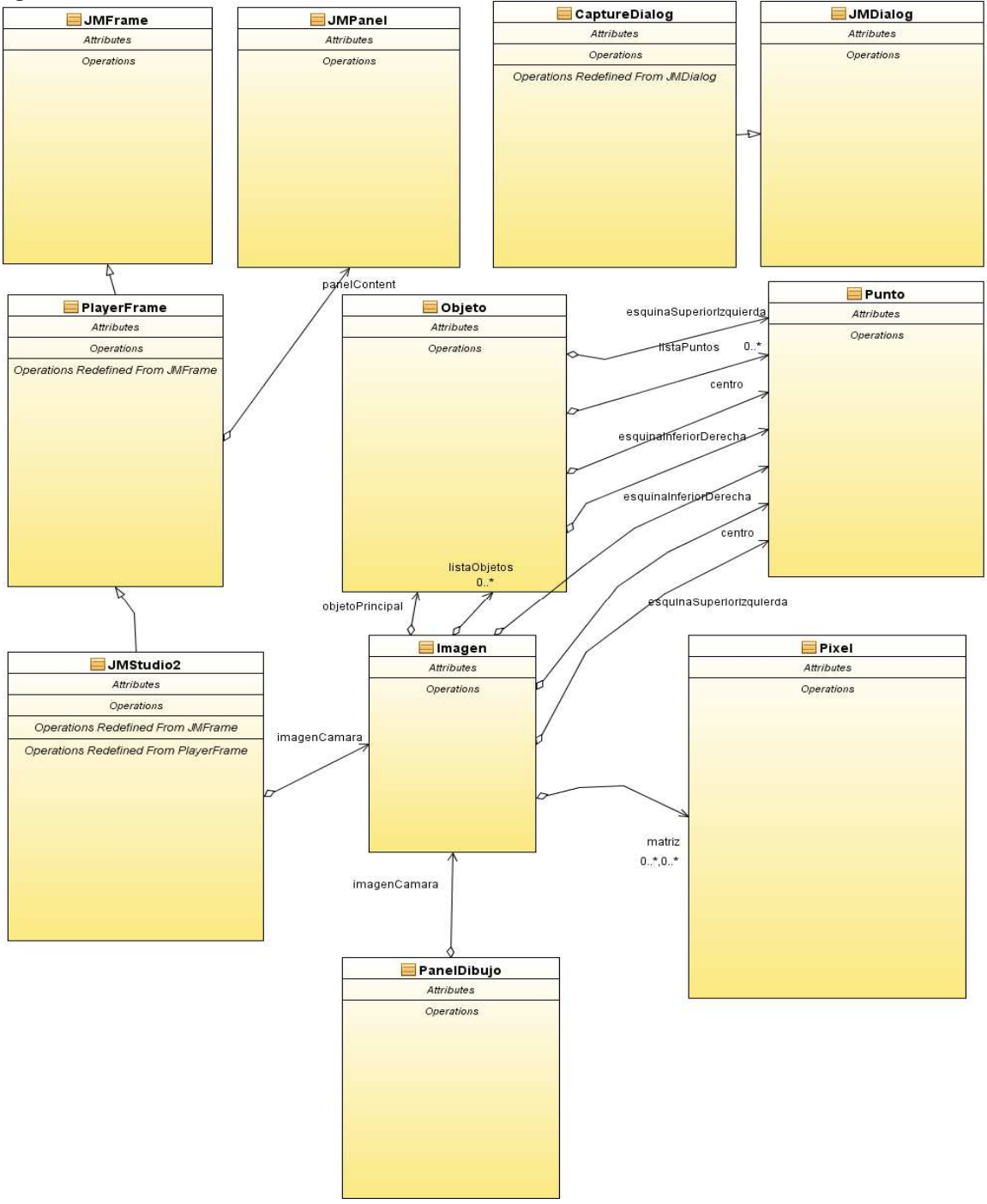


Figura 4: Diagrama de clases: Detección de la trayectoria del objeto marcado.

Por razones de espacio anteriormente no se muestran los atributos y operaciones de las clases, por lo que a continuación se detallan, una a una.

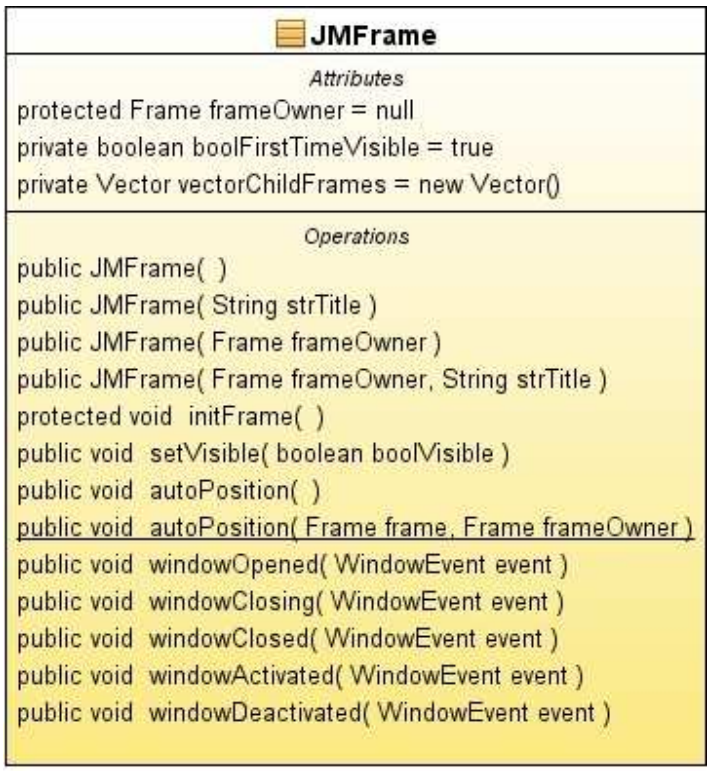


Figura 5: Diagrama de la clase JMFrame.

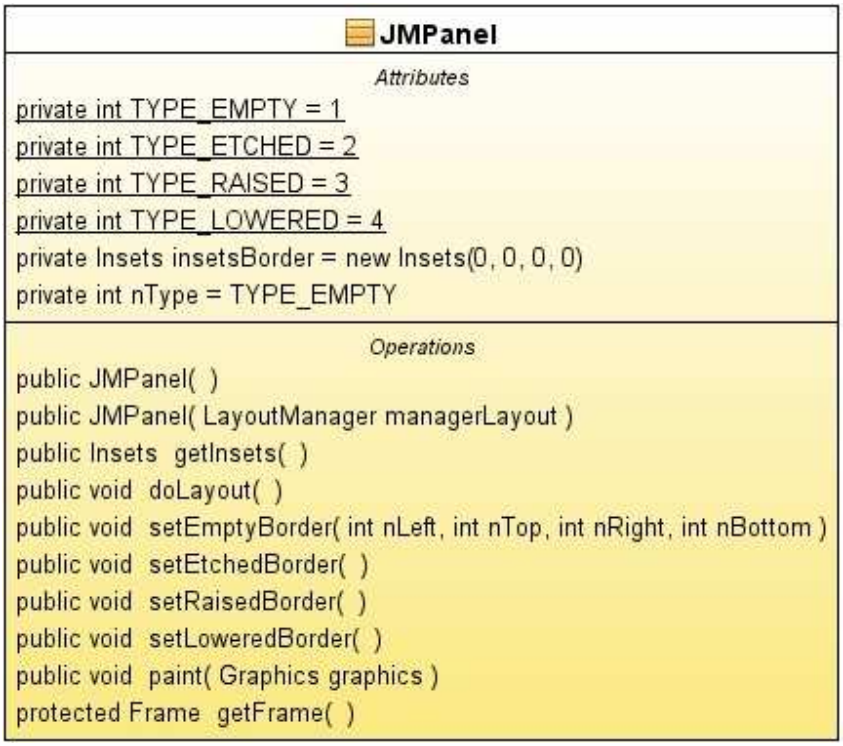


Figura 6: Diagrama de la clase JMPanel

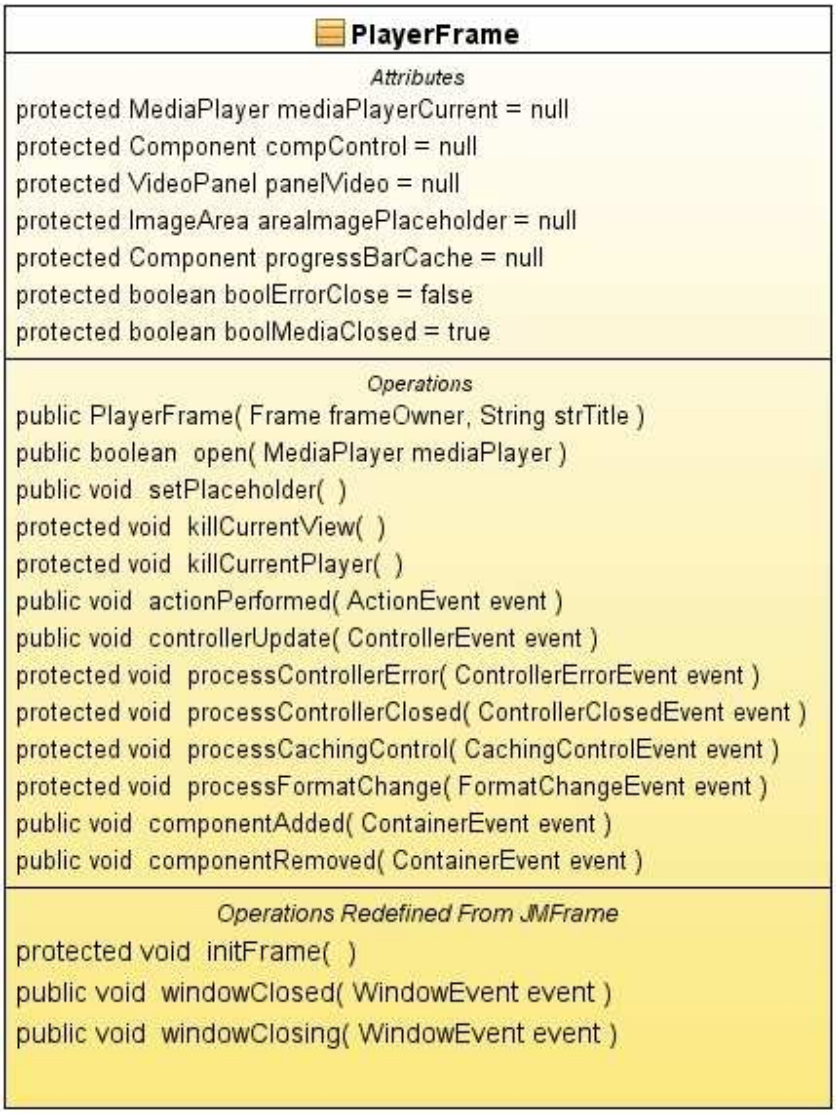


Figura 7: Diagrama de la clase PlayerFrame

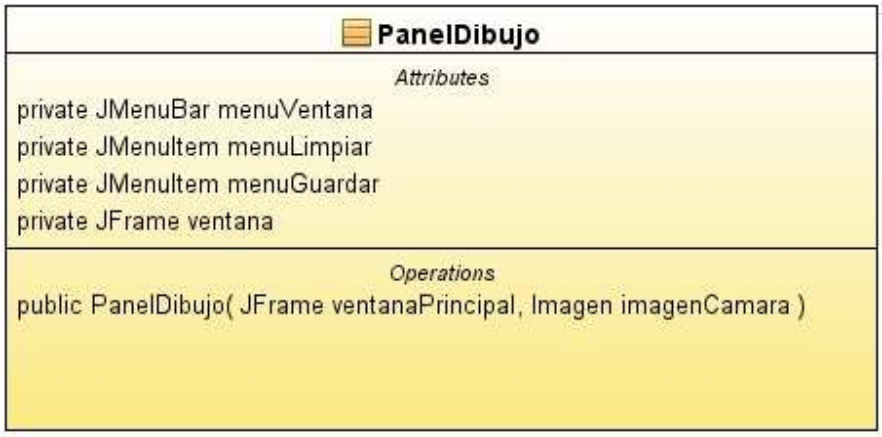


Figura 8: Diagrama de la clase PanelDibujo

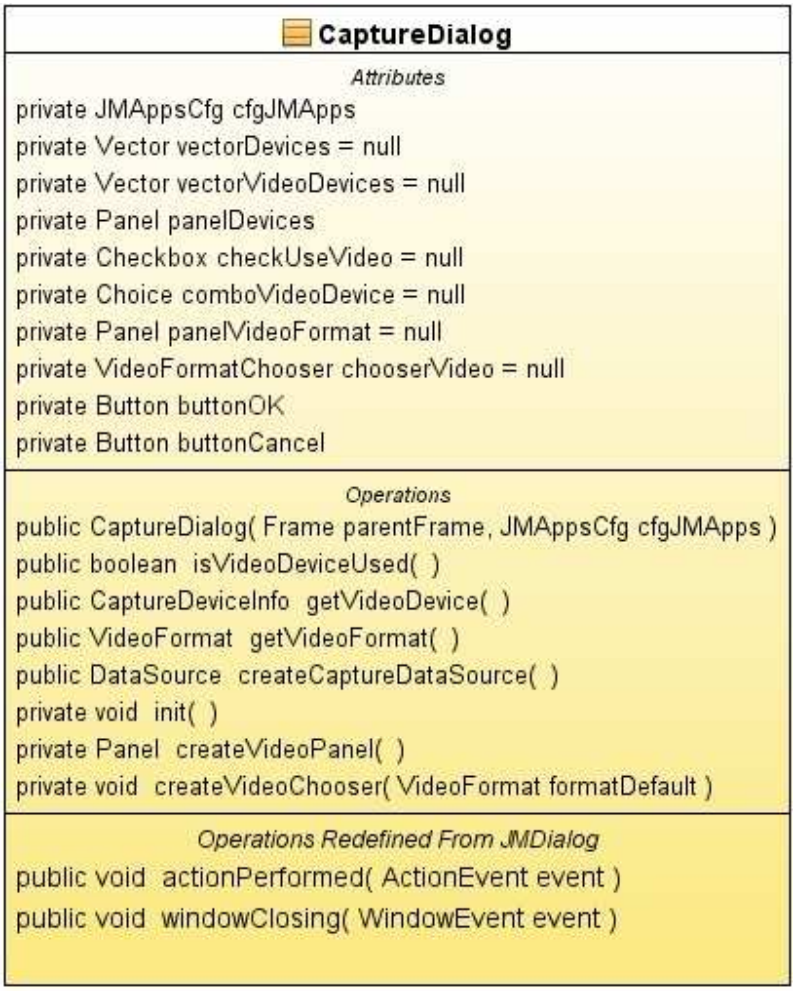


Figura 8: Diagrama de la clase CaptureDialog

Figura 9: Diagrama de la clase JMDialog

Figura 10: Diagrama de la clase Objeto

Figura 11: Diagrama de la clase Punto

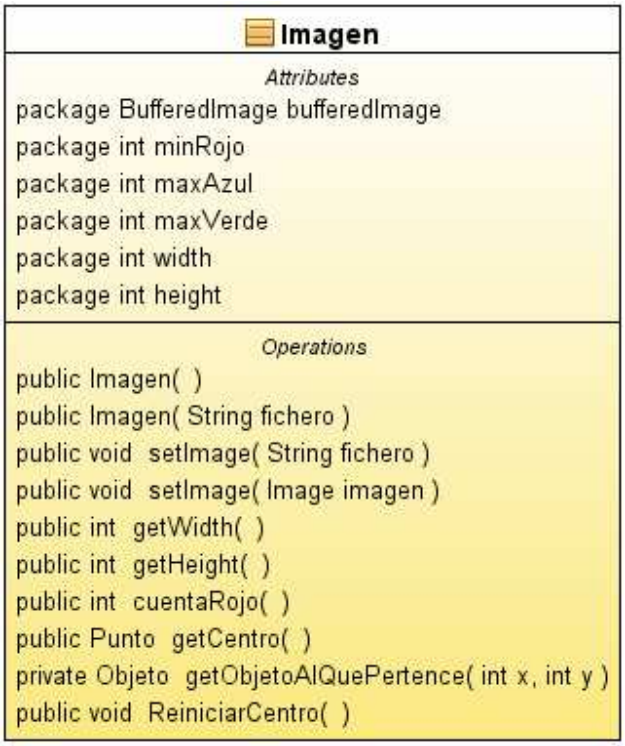


Figura 12: Diagrama de la clase Imagen

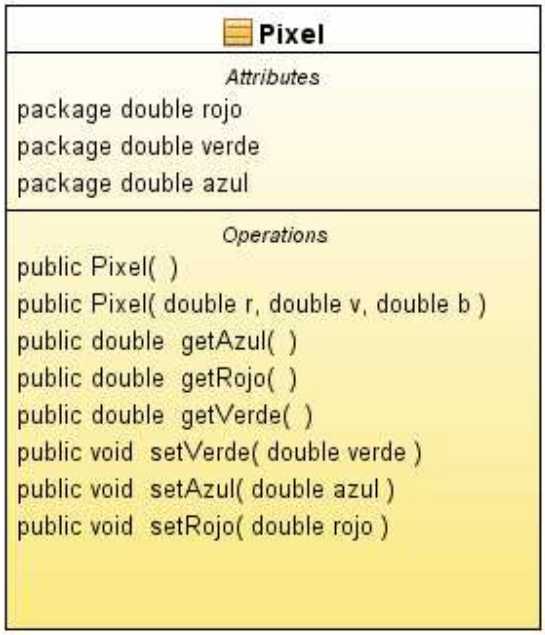


Figura 12: Diagrama de la clase Pixel

JMStudio2

Attributes

private MenuItem menuCapture
private MenuItem menuFullScreen
private MenuItem menuSnapShot
private MenuItem menuPlugins
private MenuItem menuCaptureControl
private Dimension dimFrameSizeBeforeFullScreen = null
private Window windowFullScreen = null
private MouseListener listenerMouseFullScreen
private Control controlPlugins = null
private FrameGrabbingControl controlGrabber = null
private FileDialog dlgOpenFile = null
private JMFRegistry jmfRegistry = null
private SnapFrame frameSnap = null
private DataSource dataSourceCurrent = null
private String nameCaptureDeviceVideo = null
private String videoEffect = null
private CaptureControlsDialog dlgCaptureControls = null
package Image image
package boolean killed = false
private String MENU_FILE_PREFERENCES = JMFI18N.getResource("jmstudio.menu.file.preferences")
private String MENU_FILE_EXIT = JMFI18N.getResource("jmstudio.menu.file.exit")
private String MENU_PLAYER = JMFI18N.getResource("jmstudio.menu.player")
private String MENU_PLAYER_AUTOPLAY = JMFI18N.getResource("jmstudio.menu.player.autoplay")
private String MENU_PLAYER_AUTOLOOP = JMFI18N.getResource("jmstudio.menu.player.autoloop")
private String MENU_PLAYER_KEEPASPECT = JMFI18N.getResource("jmstudio.menu.player.keepaspect")
private String MENU_PLAYER_FULLSCREEN = JMFI18N.getResource("jmstudio.menu.player.fullscreen")
private String MENU_PLAYER_SNAPSHOT = JMFI18N.getResource("jmstudio.menu.player.snapshot")
private String MENU_PLAYER_PLUGINS = JMFI18N.getResource("jmstudio.menu.player.plugins")
private String MENU_PLAYER_CAPTURE = JMFI18N.getResource("jmstudio.menu.player.capturecontrols")
private Vector vectorFrames = new Vector()
private JMAppsCfg cfgJMApps = null
private double dDefaultScale = 1.0
private Timer timer
private JPanel panelnuestro
private JFrame ventana
private JMenuBar menuVentana
private JMenuItem menuLimpiar
private JMenuItem menuGuardar
private JPanel panelDibujo
package Graphics grafico
private int centroAnteriorImagen[0..*]

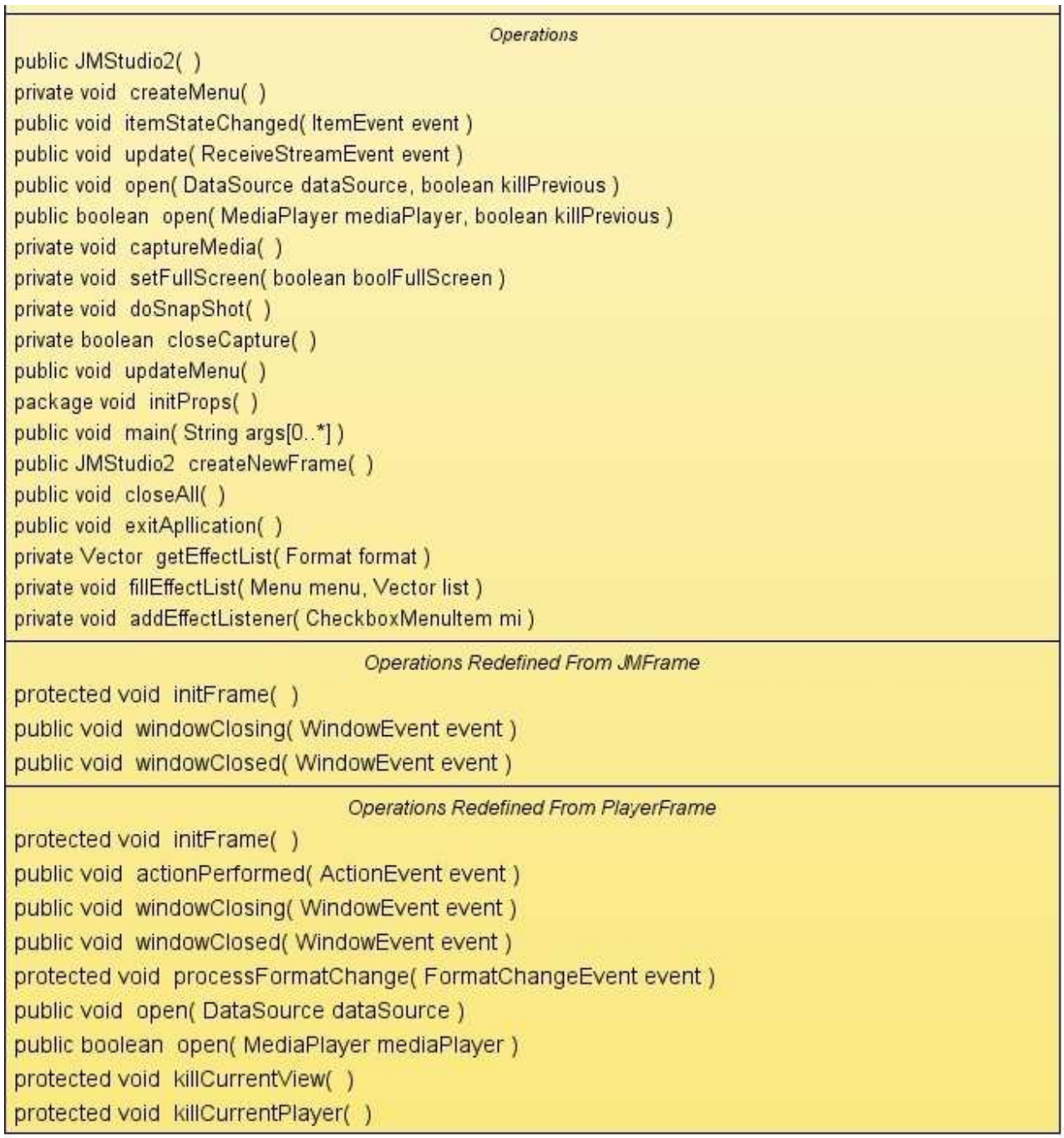


Figura 12: Diagrama de la clase JMStudio2

Ejecución de la aplicación

A través de una serie de capturas de pantalla, se intenta mostrar la ejecución del programa.
La imagen a continuación sería la pantalla principal que aparece al ejecutar la aplicación.

Cuenta con un menú File en el que se despliegan dos opciones, la primera (Capture) permite codificar, modificar o personalizar las características de la captura de video. La segunda (Exit) como es esperable, permite terminar la ejecución del programa.

Figura 13: Menú de la pantalla inicial

A continuación se muestra la ventana que aparece luego de seleccionar la opción Capture del menú.

En ella se muestran las opciones de codificación para el video, el tamaño del mismo y la frecuencia con a que se refresca la imagen del video.

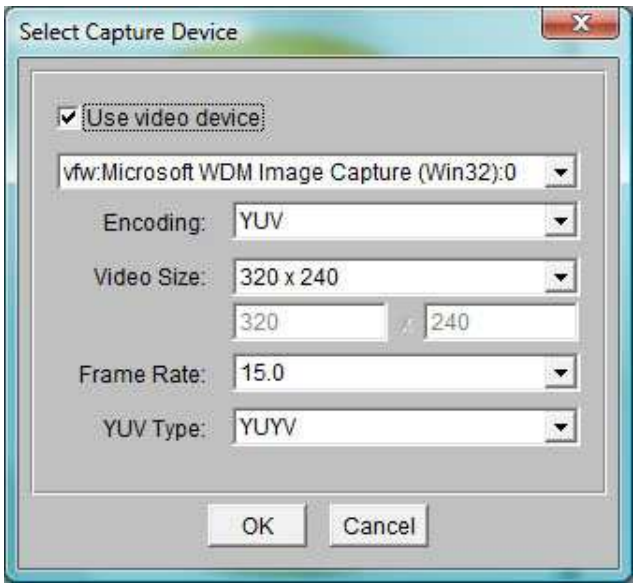


Figura 14: Formulario de selección de opciones

La opción Use video device (uso del dispositivo de video) debe estar seleccionada.

La codificación YUV (Encoding: YUV) es la única utilizada. YUV es un método que define a una señal de video que separa los componentes de luminosidad (Y) y color (UV). Individualmente, las letras YUV significan Intensity, Hue, y value. (Intensidad, matiz y valor).

La visión humana es mucho más sensitiva a las variaciones de intensidad que a las variaciones de color. El proceso de codificación YUV toma ventajas de este fenómeno y provee un ancho de banda mayor para la información de luminancia que para la de crominancia.

En la siguiente imagen también, se ve como podría variar el tamaño de video.

Las opciones del tamaño del video así como la frecuencia de refresco influyen directamente en la velocidad de ejecución de la aplicación.

En este caso es importante tener en cuenta que al aumentar el tamaño de la imagen del video, aumenta el número de píxeles a tratar posteriormente por cada imagen, lo que determina la respuesta de la aplicación.

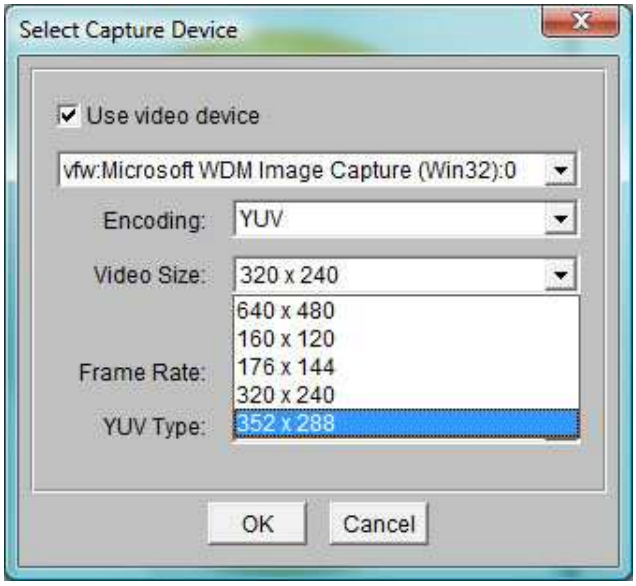


Figura 15: Selección del tamaño del video

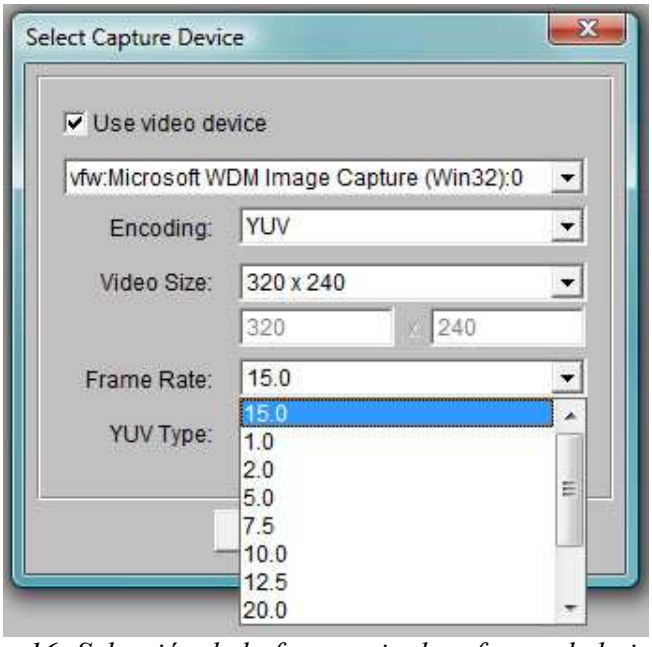


Figura 16: Selección de la frecuencia de refresco de la imagen

Finalmente se muestra como es la ejecución del programa en esta fase. Por un lado esta el formulario donde se dibuja la trayectoria del objeto tratado y por otro lado se muestra la captura de la WebCam en ese momento.

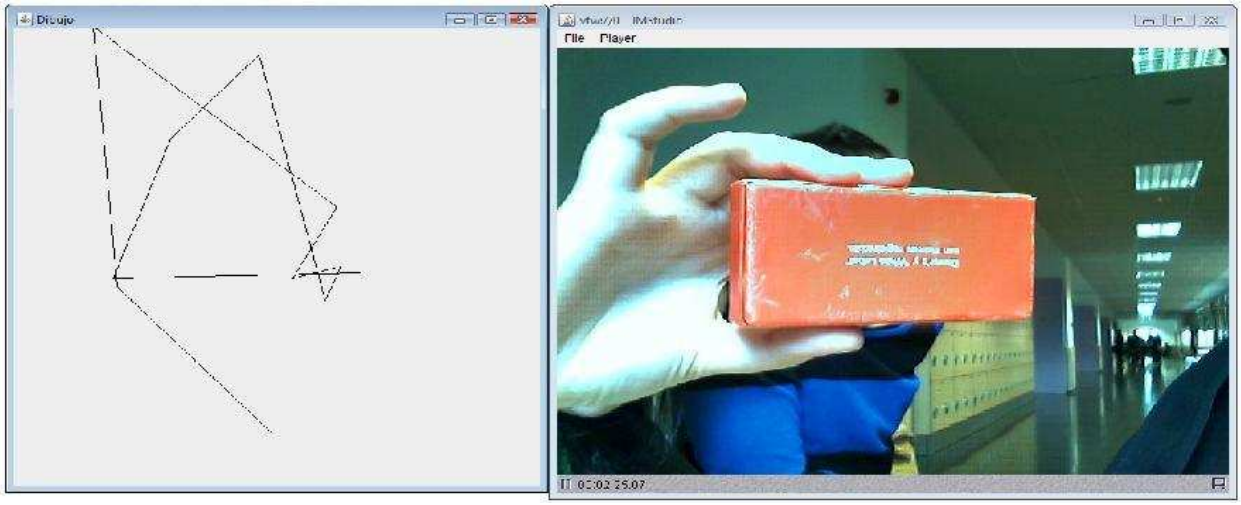


Figura 17: Muestra de la simulación del programa

Figura 18: Muestra de la simulación del programa

El formulario donde se dibuja la trayectoria del objeto "rojo" cuenta a su vez con la posibilidad de guardar como una imagen en formato .jpg esa trayectoria capturada.

Aquí se muestra un ejemplo de la trayectoria del objeto dibujada sobre el formulario.

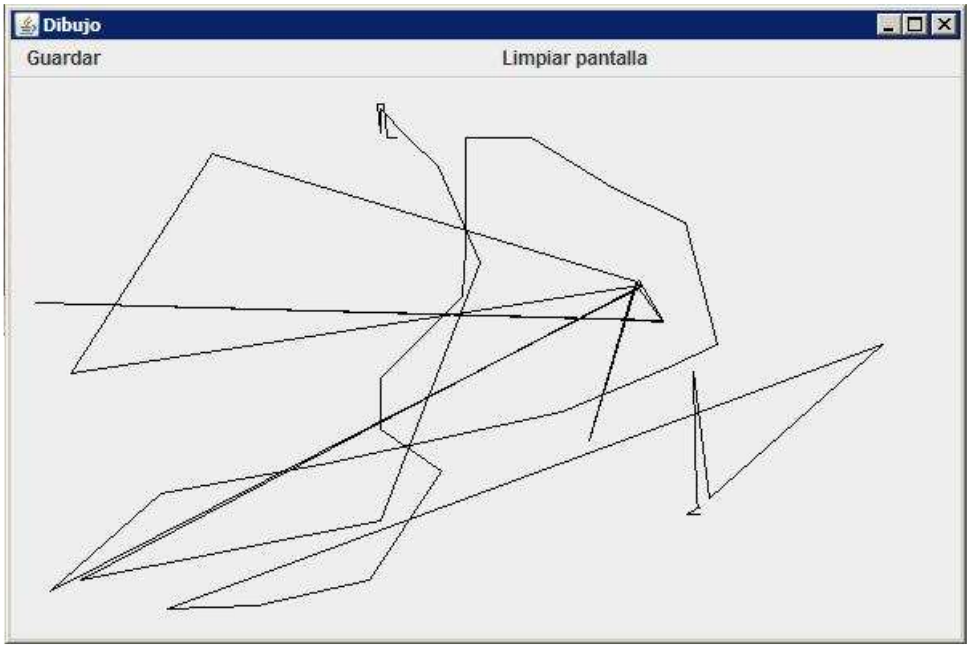


Figura 19: Ejemplo de la trayectoria dibujada en el formulario gráfico

Tras pulsar opción *Guardar* del Menú del formulario, se guarda la imagen en la carpeta del proyecto en un archivo llamado "foto.jpg". En este ejemplo este es el resultado:

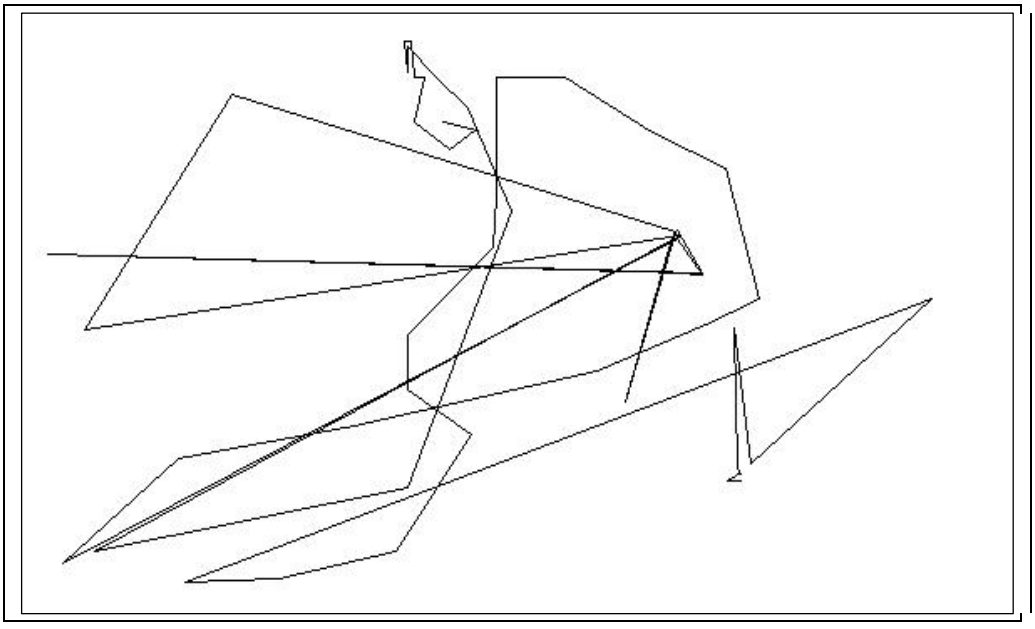


Figura 20: Archivo "foto.jpg"

Para finalizar, también se ofrece la posibilidad de limpiar la pantalla para comenzar nuevamente a dibujar el trazo que representa la trayectoria del objeto tratado.

Figura 21: Muestra opción "Limpiar pantalla" del formulario gráfico

2.2.2 Manejo de un escritorio virtual (Simulación del clic del ratón)

Breve explicación

En principio, se quiere que la propia imagen del entorno real en el que se encuentra el usuario sea como el fondo de escritorio de cualquier ordenador convencional. De esta forma, el usuario podrá seleccionar una carpeta de su escritorio, para desplazarla a otro sitio de la pantalla. Para ello, el usuario tendrá que desplazar el objeto marcado del proyecto hasta la carpeta en cuestión, hacer clic sobre ella y posteriormente escoger una nueva posición sobre la pantalla para hacer clic sobre ella y mover así la carpeta a esa nueva posición.

Desarrollo

Partiendo de lo conseguido en apartado 2.2.1 *Detección de la trayectoria del objeto marcado*, en donde la detección de patrón estaba conseguida, se buscó implementar una especie de clic de ratón, donde lo fundamental a tener en cuenta fué, como representar un clic, y fijar un margen de tiempo para que el evento tuviera sentido.

La manera de simular el clic se basa en el tratamiento de la imagen capturada por la WebCam. Durante la ejecución determinamos el área formada por el objeto rojo analizado y cuando se observa un cambio notable del área determinada por el umbral escogido, lo determinamos como si fuera hacer un clic sobre la pantalla del formulario.

Para esta aplicación utilizamos cuatro imágenes en total. Usamos una para simular el fondo del escritorio de un ordenador convencional, otra para simular que el movimiento del objeto analizado es como el puntero de un ratón y tras dos para diferenciar el estado de la carpeta que vamos a desplazar; esta diferencia radica en que si aún no hemos hecho clic sobre la carpeta, se muestra la imagen de una carpeta marrón sobre un fondo azul marino y en cuanto hacemos clic sobre ella, observamos una imagen diferente de color naranja.

El uso de la aplicación es muy intuitivo pues en todo momento podemos saber en la posición en la que nos encontramos con sólo observar en que posición se encuentra imagen del puntero del ratón.

Partes de la aplicación

Lo principal en este caso es la implementación o la simulación del clic del ratón. Para ello dentro de la función CaptureMedia en la clase JMStudio2, se añade un actionListener asociado al timer principal de manera que si dentro de un intervalo de tiempo se producen dos clics, se interprete como tal y se prepare para desplazar el objeto a otro sitio del escritorio una vez que se vuelva a detectar dos clics en otra zona del escritorio, y de este modo se desplace el objeto de su posición inicial a esta última donde se ha hecho el clic.

Los clics son interpretados como un aumento proporcional de la cantidad de píxeles rojos que representa el objeto principal. Si dentro del intervalo de tiempo marcado por el Timer, aumenta la proporción de píxeles rojos dos veces consecutivas, se lleva a cabo el clic.

La parte gráfica que muestra el resultado de la acción del clic está implementada mediante el método drawImage que ofrece la api de Java para el JFrame o Fomulario gráfico, de manera que de acuerdo a lo que corresponda, se dibuje una imagen u otra en el formulario.

La importancia de esta aplicación radica en la detección del clic y la carga y modificación de las imágenes que aparecen el la pantalla en función de los clics y desplazamientos que hagamos con la carpeta que movemos en el formulario.

```
//Se distingue si ha sido seleccionado o no para dibujar imágenes diferentes

    if(!seleccionado){
      if(posRecX2 >=0){
        if(posRecY2 >=0){

panelDibujo.getGraphics().drawImage(carpeta,posRecX,posRecY,anchoRec,alto
Rec, null);
        }else{

panelDibujo.getGraphics().drawImage(carpeta,50,50,anchoRec,altoRec,
null);
        }
        }else{
            if(posRecY2 <0){

panelDibujo.getGraphics().drawImage(carpeta,posRecX,posRecY,anchoRec,alto
Rec, null);

                        }
                }
            }

                else{

                    if(posRecX2 >=0){
                        if(posRecY2 >=0){

panelDibujo.getGraphics().drawImage(carpeta2,posRecX,posRecY,anchoRec,alt
oRec, null);

                            }else{

panelDibujo.getGraphics().drawImage(carpeta2,50,50,anchoRec,altoRec,
null);
                                }

                                }else{
                                if(posRecY2 <0){
```

```
panelDibujo.getGraphics().drawImage(carpeta2,posRecX,posRecY,anchoRec,altoRec, null);

                                                    }
                                              }

                                        }

 // Al final del todo va el cursor, de manera que se sobre ponga al fondo y a la imagen de la carpeta.


      panelDibujo.getGraphics().drawImage(flecha,(int)(panelDibujo.getWidth() -
centro.getX()*relacionVentanaImagenX),(int)(centro.getY()*relacionVentanaImagenY),30,30, null);
                                        centroAnteriorImagen[0]=centro.getX();
                                        centroAnteriorImagen[1]=centro.getY();
                                  }



       int area=0;
        if(imagenCamara != null)
            area = imagenCamara.getArea();

            double diferencia = 1.8;

            if(((area-10) > areaAnterior*diferencia){
                  if(vecesVariacionArea == 1){
                        vecesVariacionArea = 0;
                        time2Avanza = false;
                        if(seleccionado){
                              posRecX=panelDibujo.getWidth() -
((int)(centro.getX()*relacionVentanaImagenX));//-posRecX);

      posRecY=(int)(centro.getY()*relacionVentanaImagenY);//-posRecY;
                                        seleccionado=false;
                                        }

                                  }else{

                                        time2Avanza = true;
                                        int aux,aux2;

                                        aux=panelDibujo.getWidth() -
(int)(centro.getX()*relacionVentanaImagenX);

      aux2=(int)(centro.getY()*relacionVentanaImagenY);

        //Aquí vamos a comprobar si hemos clickeado sobre la superficie del rectángulo
        if(estaDentro(aux,aux2)){
            seleccionado=true;
            vecesVariacionArea = vecesVariacionArea + 1;
            }
```

```
}
 time2 = 0;

        }

        areaAnterior = area;


              if(time2Avanza)
                    time2++;
                     int tiempoLimite = 5000/timer3;

                     if(time2 >= tiempoLimite){
                          vecesVariacionArea=0;
                          time2Avanza = false;
                          time2 = 0;
                    }
```

UML

Se añaden las clases ColaImagenes y MetaImagenes y se ve modificada la función CaptureMedia() de la clase JMStudio2 pues se añade la captura del clic y cambia el actionListener asociado al Timer.

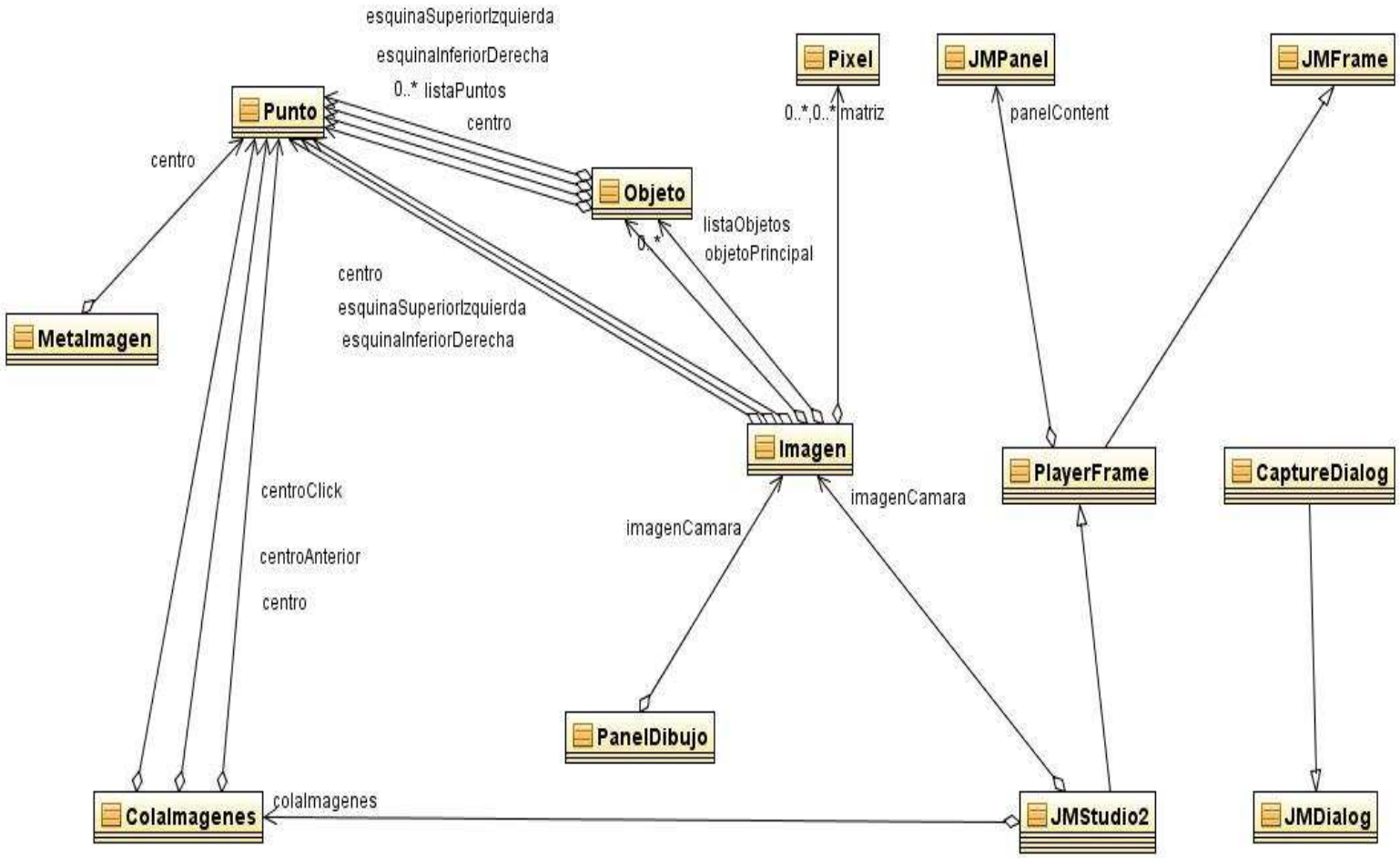


Figura 22: Diagrama de clases: Escritorio Virtual

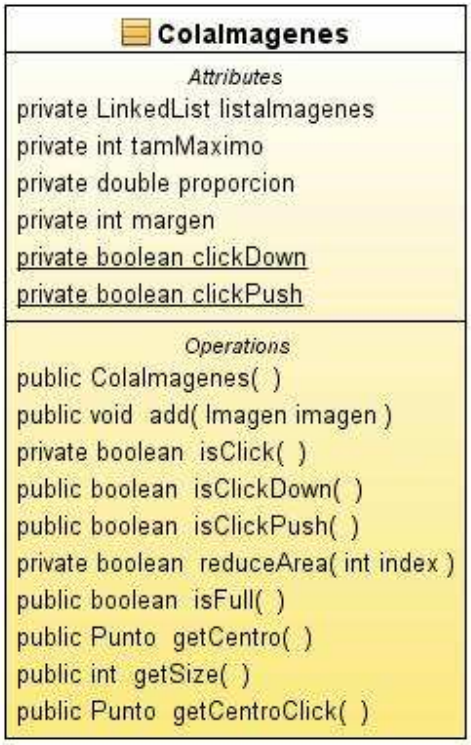


Figura 23: Diagrama de la clase ColaImagenes

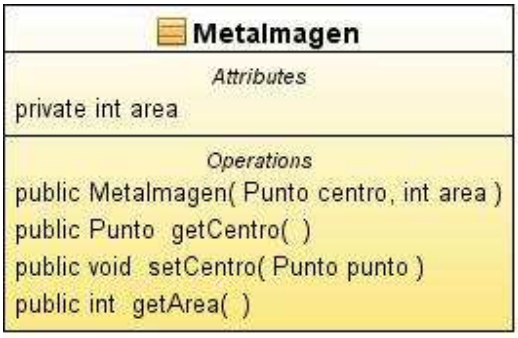


Figura 24: Diagrama de la clase MetaImagen

Ejecución de la aplicación

Ahora se muestra visualmente el uso de la aplicación detallada paso por paso. Las imágenes corresponden a una ejecución representativa del escritorio virtual.

En esta primera imagen se observa como se inicia la aplicación. La carpeta se encuentra en la parte superior de la pantalla y vamos a desplazarnos hasta ella para hacer un clic.

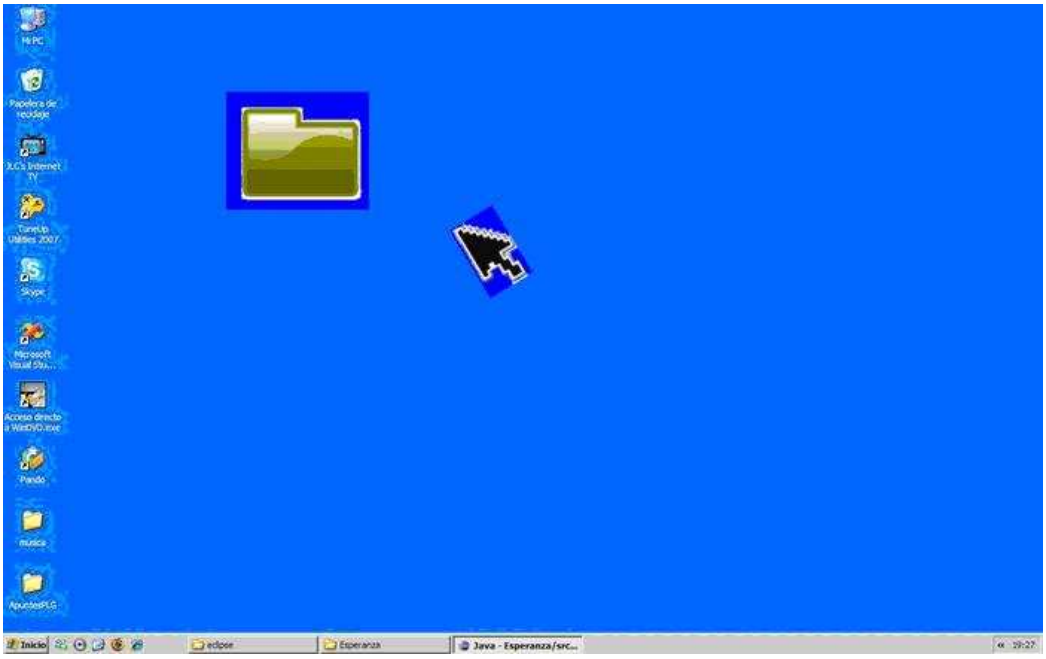


Figura 25: Escritorio virtual

Como se ha dicho anteriormente, la imagen de la carpeta cambia tras haber hecho clic sobre ella. Ahora nos desplazamos a una nueva posición donde se va a hacer un segundo clic y así poner la carpeta en una nueva posición.

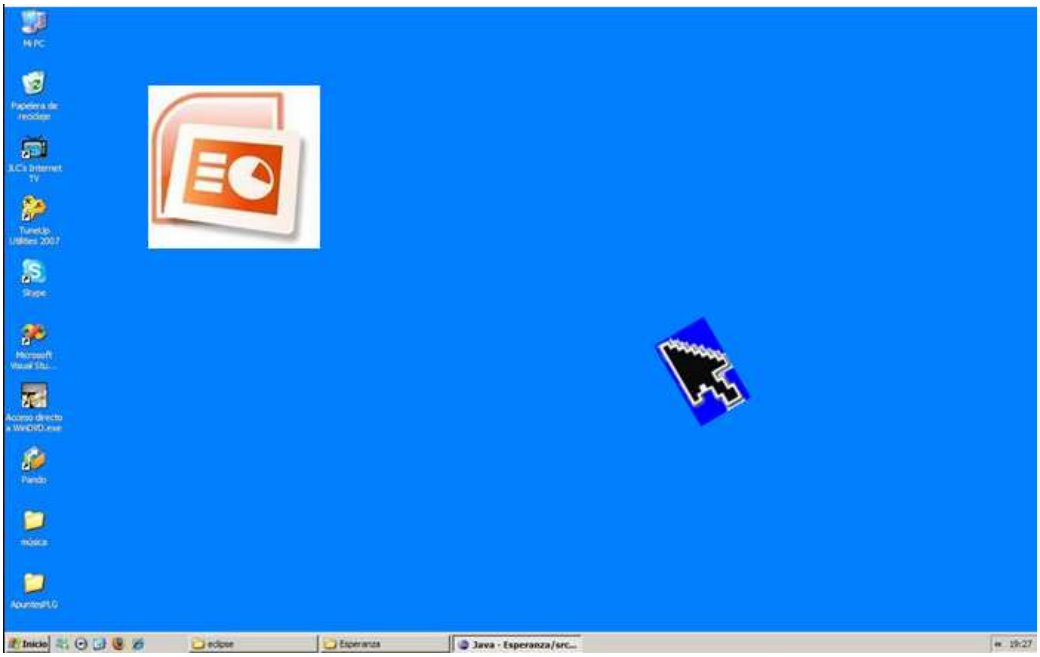


Figura 26: Clic sobre la carpeta

Como se observa en la siguiente imagen, la carpeta se ha situado en una nueva posición del formulario y la imagen también ha cambiado pues se encuentra en el estado de que aún no se ha hecho clic sobre ella en ese nuevo lugar.

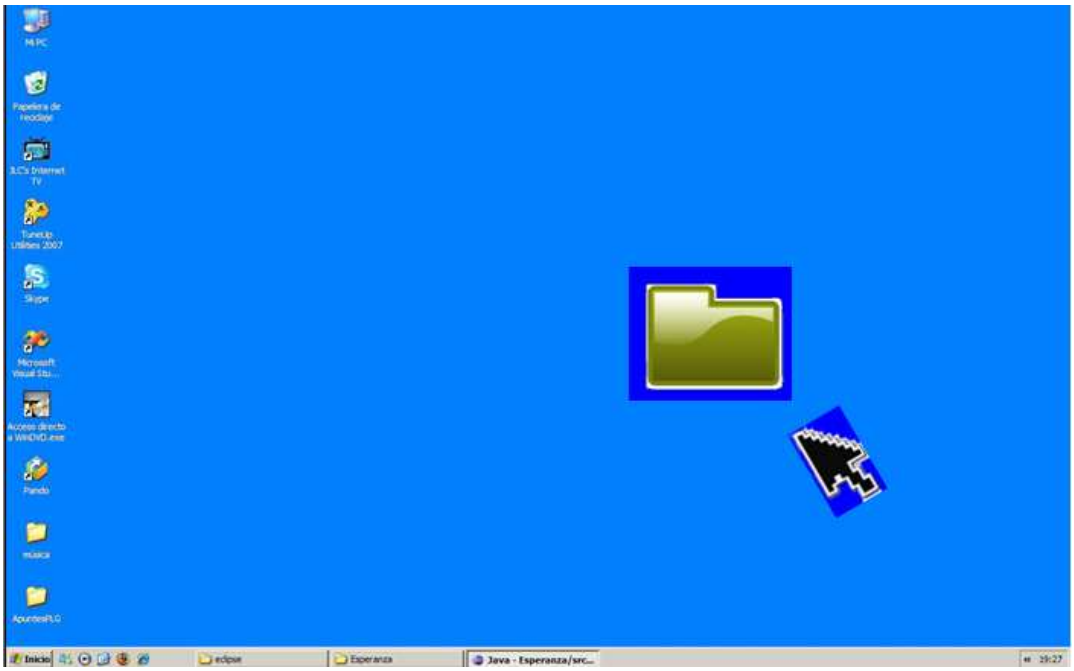


Figura 27: Desplazamiento de la carpeta a una nueva posición

2.2.3 Teléfono inteligente

Breve explicación

Esta aplicación simula el funcionamiento de un teléfono cotidiano manejado a través de la herramienta creada en el proyecto. Se puede marcar un número de teléfono pulsando las teclas de una en una, como se haría en un teléfono normal y posteriormente pulsar la tecla de llamada simulando el establecimiento de llamada al número marcado anteriormente. Además se puede en cualquier momento detener la acción realizada pulsando sobre la tecla colgar para poder comenzar de nuevo con la simulación.

Desarrollo

En esta aplicación también partimos de la herramienta creada con anterioridad para detectar la trayectoria del objeto marcado y con la detección del clic para determinar la selección de la tecla pulsada sobre la imagen que el usuario visualiza mientras usa la simulación del teléfono.

Para desarrollar esta aplicación se dibuja sobre el formulario las 12 teclas más comunes de todos los teléfonos, diez botones para los números del 0 al 9 y otros dos para realizar las acciones de llamar y colgar. Estas imágenes se centran en el formulario para una mejor visualización del dispositivo y para determinar con exactitud las pulsaciones realizadas con el objeto marcado. En la parte superior del teclado se escribe progresivamente la concatenación de las teclas que el usuario marca y mostramos mensajes correspondientes al establecimiento de la llamada o al momento en el que el usuario cuelga para volver a marcar un nuevo número de teléfono.

Se han usado imágenes correspondientes a cada número del teléfono y adecuadas para intuir la acción de cada tecla con sencillez, del mismo modo se añade la tecla de llamar representada con el dibujo del aparato en cuestión de color verde y la tecla de colgar exactamente igual pero de color naranja.

Para una mejor visualización de la tecla presionada, hemos creado también imágenes para mostrar un cambio en el botón pulsado por el usuario y así aumentar la simplicidad de uso de la aplicación ya que en todo momento puedes conocer el botón que ha sido marcado la última vez que se hizo clic sobre una tecla del formulario que se tiene delante.

Partes de la aplicación

Para crear esta aplicación se ha modificado la clase principal del proyecto y añadido otras nuevas para facilitar su ejecución.

Ahora se cuenta con una **nueva clase llamada Botón** para tener las propiedades que caracterizan las teclas del teléfono. Entre sus atributos se encuentran: las coordenadas X e Y de la esquina superior izquierda donde se dibujará el botón, los String que determinan la ruta origen de las imágenes que mostrarán al botón pulsado o no pulsado, el entero que corresponde con el número del botón y un atributo booleano para conocer el estado en el que se encuentra la tecla (pulsado o no pulsado).

En cuanto a los **cambios realizados en la clase JMStudio2**, se ha cambiado fundamentalmente las funciones que se ejecutan en el actionListener asociado al timer principal para adecuarlas a lo que necesitamos generar en esta aplicación. También se han añadido nuevos atributos para guardar y controlar las funciones del teclado: un arrayList de Botones para controlar el uso de los mismo, un String que se visualizará en todo momento en el formulario para controlar el uso de la aplicación y los valores adecuados del ancho y del largo de los botones para mostrarlos en función del ancho y largo del formulario general en el que se van a visualizar.

En esta ocasión, cada nueva ejecución del actionListener recorrerá el arrayList de Botones para generar uno a uno las teclas del teléfono y mostrará la cadena llamada *numeroMarcado* que controla la acción que se está realizando en un momento preciso. Además se tiene en cuenta el atributo pulsado de cada uno de los botones para escoger la imagen que se visualizará de cada uno de ellos y actualizaremos la cadena dicha con anterioridad según los siguientes conceptos:

-Si se está marcando el número de teléfono al que se quiere llamar, se irán concatenando en el String cada uno de los botones pulsados por el usuario.

-Una vez que se hayan marcado todos los números que componen el número de la llamada, se hace clic sobre la tecla “Llamar” y la cadena mostrada por pantalla se modificará poniendo en su lugar la frase “Estableciendo llamada...”. En ese preciso instante ya no se puede pulsar ninguna otra tecla numérica, solamente se tiene la opción de hacer clic sobre colgar para finalizar la simulación de la llamada y volver a escoger los números a marcar.

-Como se ha dicho anteriormente, el botón “Colgar” se pulsará para finalizar el establecimiento de una llamada. Además este botón es adecuado por si se pulsa erróneamente una tecla durante la selección del teléfono al que vamos a llamar. Tras pulsarlo, la cadena numeroMarcado cambiará “Llamada finalizada Vuelva a marcar” y se podrá comenzar de nuevo con la selección de las teclas a pulsar.

La importancia de este método también radica en la detección del clic además de la modificación de las imágenes que aparecen en el formulario según las teclas pulsadas del teléfono y el texto que determina el estado en el que se encuentra la llamada en todo momento. Por tanto se muestra un extracto de código relevante.

```
for (int i=0;i<12;i++){


      File ruta=null;
      Image im=null;
      if(!teclado[i].isPulsado())
            ruta= new File(teclado[i].getRutaImagen());
      else
            ruta= new File(teclado[i].getRutaModificado());

            try {
                  im = ImageIO.read(ruta);
            }
            e1.printStackTrace();
 }
panelDibujo.getGraphics().drawImage(im,
teclado[i].getX(),teclado[i].getY(),anchoTecla, altoTecla,null);
                                    }

      Font fuente=new Font("TimesRoman", Font.BOLD, 12);
            //CASO ESPECIAL BOTON LLAMAR
      panelDibujo.getGraphics().setFont(fuente);


        //AQUI ESCRIBIMOS EL NUMERO MARCADO O EL ESTADO DEL TELEFONO
panelDibujo.getGraphics().drawString(NumeroMarcado,anchoTecla*2,altoTecla
/2);

      centroAnteriorImagen[0]=centro.getX();
      centroAnteriorImagen[1]=centro.getY();
      }
      int area=0;
      if(imagenCamara != null)
            area = imagenCamara.getArea();
            double diferencia = 1.8;
  // Para contar si es un click tenenemos que controlar que el centro sea
mas o menos el mismo que el anterior

        if(((area-10) > areaAnterior*diferencia)

        time2Avanza = true;
        int aux,aux2;

    aux=panelDibujo.getWidth() -
(int)(centro.getX()*relacionVentanaImagenX);

      aux2=(int)(centro.getY()*relacionVentanaImagenY);

//Aqui vamos a comprobar si hemos clickeado sobre la superficie del
rectángulo

      //Calculamos la tecla pulsada
      int tecla=calculaTecla(aux,aux2);

//Controlamos que hemos hecho un clic sobre alguna tecla
      if(tecla!=-1){
            Font fuente=new Font("TimesRoman", Font.BOLD, 20);
            //CASO ESPECIAL BOTON LLAMAR
```

```
panelDibujo.getGraphics().setFont(fuente);

//Controlamos que hemos pulsado una tecla numerica o que no estamos
realizando la llamada

      if((tecla<10)&&(!NumeroMarcado.equals("Estableciendo
llamada....")){
        cambiaTecla(tecla);

        //Comprobamos que no se está haciendo la conexión
            if(!NumeroMarcado.equals("Llamada finalizada vuelva a
marcar")){

      NumeroMarcado=NumeroMarcado+" "+Integer.toString(tecla);
        }
            else{
            //Inicio de la pulsacion de teclas
            NumeroMarcado=Integer.toString(tecla);
            }
        }
//Controlamos que estamos pulsando una tecla de acción telefónica
        else{
            cambiaTecla(tecla);
            if (tecla==10)
                  NumeroMarcado="Estableciendo llamada....";

            if (tecla==11)
                  NumeroMarcado="Llamada  finalizada  vuelva  a  marcar";
```

UML

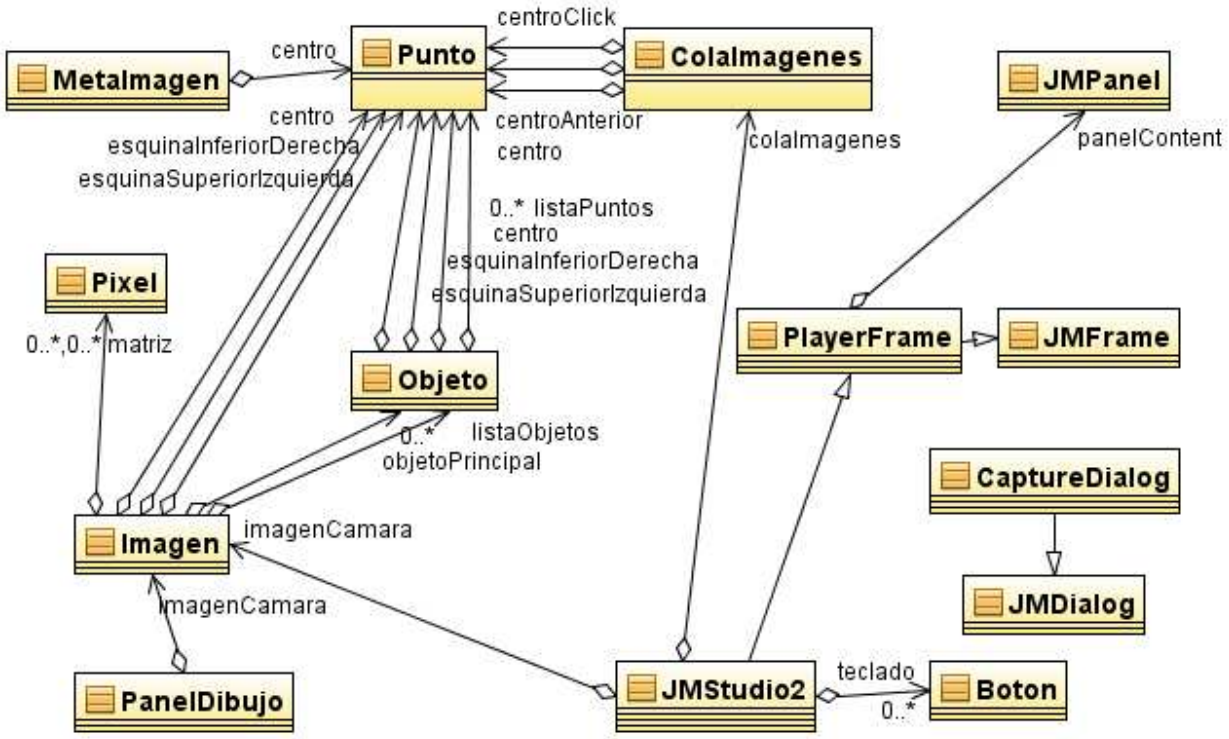


Figura 28: Diagrama de clases Teléfono Inteligente

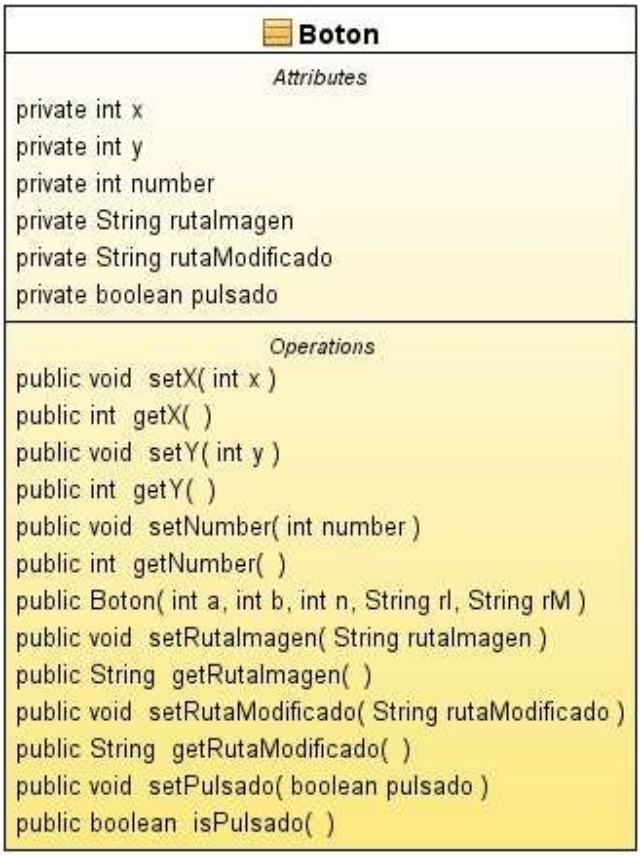


Figura 29: Diagrama de la clase Boton

Figura 30: Parte 1 Diagrama de la clase JMStudio2

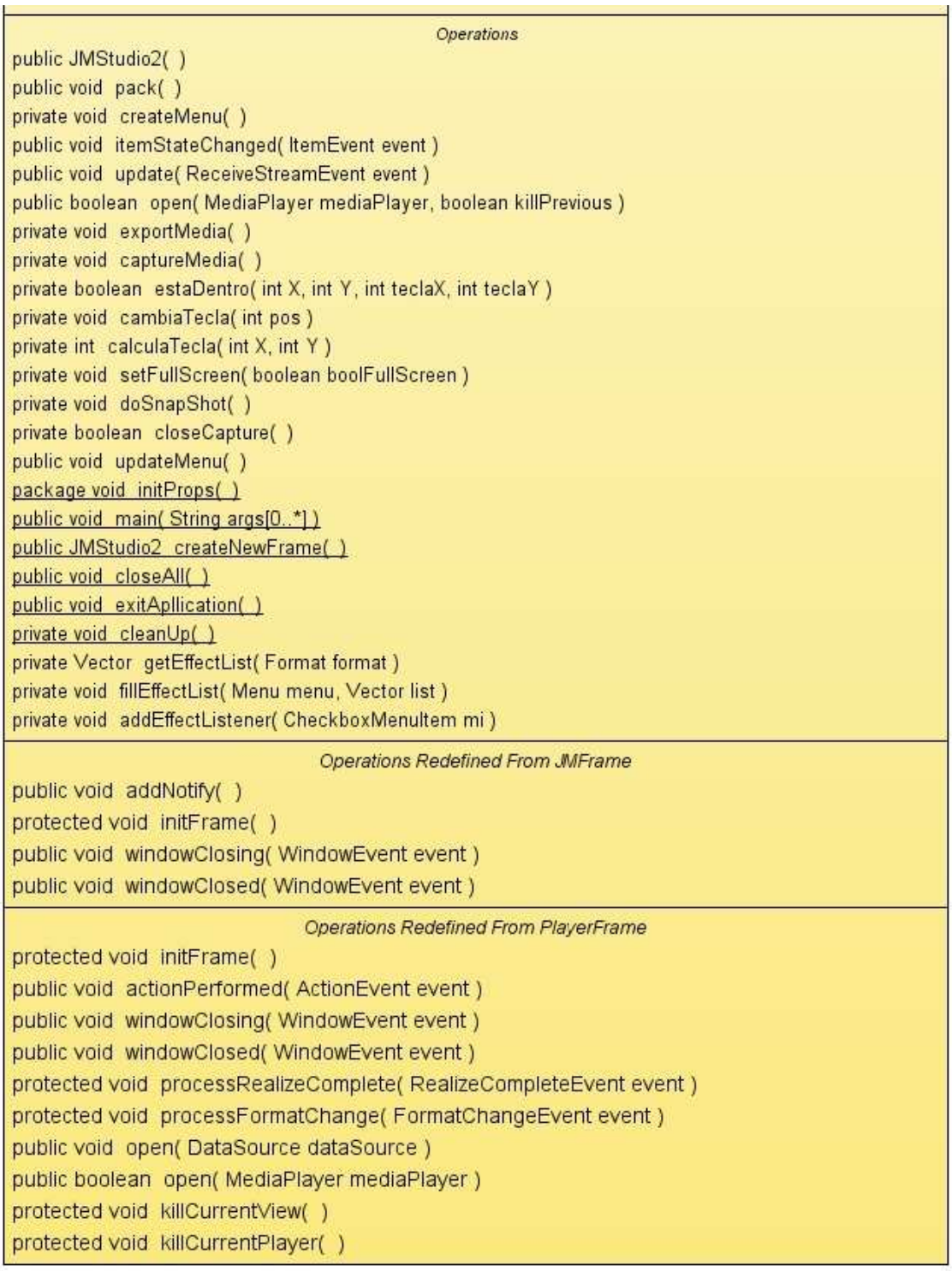


Figura 31: Parte 2 Diagrama de la clase JMStudio2

Ejecución de la aplicación

Ahora se muestra visualmente el uso de la aplicación detallada paso por paso. Las imágenes corresponden a una ejecución representativa del teléfono inteligente.

Esta imagen corresponde al teclado antes de haber sido pulsada alguna tecla.

Figura 32: Teclado convencional de un teléfono

Primero se muestra un ejemplo en el que se ha pulsado el número 86549 y como se puede observar en la imagen, la última tecla pulsada ha sido la correspondiente al número nueve pues se nota un aspecto diferente en la imagen que se visualiza en el teclado para ese botón y la que tienen todos los demás.

Figura 33: Tecla numérica pulsada

Ahora se visualiza un ejemplo tras haber pulsado la tecla "Llamar". Se puede observar que en parte superior del formulario se escribe "Estableciendo llamada..." en vez de los números pulsados. También se puede ver que la imagen de la tecla ha sido modificada.

Figura 34: Acción Llamar

Esta última imagen muestra el estado del formulario tras haber pulsado la tecla "Colgar". Se observa que la frase mostrada en pantalla ha cambiado y que el botón tiene otra imagen.

Figura 35: Acción colgar

2.3 Errores y problemas

Poder obtener imágenes de la WebCam y a partir de ellas poder manipularlas de forma que se pudiera obtener el reconocimiento de algún patrón determinado hemos tenido los siguientes percances:

- **Elección del lenguaje de programación**

En primer lugar se optó por desarrollar la aplicación en *Visual Studio* para poder hacer uso de las librerías SDK. Se realizaron diversas pruebas y se utilizaron códigos fuente específicos para poder conseguir la visualización de las imágenes de la cámara Web pero no se obtuvo ningún resultado productivo.

Posteriormente se decidió optar por realizar el proyecto en el lenguaje C++ porque con éste ya se tenía más experiencia pues ya se había utilizado para múltiples asignaturas de la carrera. Se utilizaron librerías útiles para la captura de imágenes a través de la cámara Web y su posterior manipulación. El problema estaba en que estas librerías no eran fácilmente adaptables a los requerimientos del programa. Se empleó mucho tiempo con este problema y después de observar que no se conseguía avanzar decidimos volver a cambiar el lenguaje de programación.

Finalmente la aplicación ha sido desarrollada en Java. Se encontró una librería que servía para presentar la imagen capturada por la WebCam a través de un formulario y poder guardar las imágenes cada cierto tiempo para su posterior manipulación.

- **Tratamiento de las imágenes**

Las imágenes capturadas están en formato .JPG

Se decidió investigar de diversas fuentes la forma de tratar las imágenes de acuerdo a lo que se buscaba. Se intentó el uso de varias librerías pero cada una de ellas planteaba nuevos problemas de formatos o de eficacia en cuanto a costes en el tiempo que no se podía permitir ya que se necesitaba reconocer patrones de varias imágenes en poco espacio de tiempo.

Finalmente se optó por crear la propia clase Píxel que guarda en una matriz de píxeles los tres valores correspondientes al RGB de cada uno de ellos (componentes de rojo, verde y azul). También se implementaron las funciones necesarias para que se pudiera seguir el reconocimiento de un patrón determinado en la matriz y así ajustar el centro de dicho patrón para poder servirnos de punto guía para los siguientes pasos en el proyecto.

En este caso, se decidió de igual forma, que el patrón sería un objeto de un determinado color que destacara, para poder reconocerlo de la forma más adecuada y fácil.

- **Visualización del punto guía**

A partir de las funciones realizadas en la clase Píxel, se decidió mostrar por pantalla los puntos concretos que nos merecía la pena observar para concretar el buen funcionamiento del reconocimiento. En un primer momento se observó que aproximadamente los valores que aparecían por pantalla eran correctos pero de vez en cuando los resultados eran completamente erróneos, por lo que nos supuso el siguiente problema.

- **Problemas con la determinación del color del patrón**

A pesar de tener correctamente guardados las componentes RGB de los píxeles, la determinación de los puntos necesarios era poco fiable en múltiples ocasiones. Se quería obtener el punto superior del objeto y el punto inferior y así, a partir de ellos, calcular su centro. Se optó crear un nuevo panel en el que poder dibujar la trayectoria del centro del objeto y así poder aclarar las dudas de los errores.

Entonces surgió otro pequeño problema**:** *los puntos dibujados no se correspondían con el tamaño de la nueva ventana*. El problema era con respecto a la relación de la pantalla en la que aparecía la imagen de la WebCam y la nueva creada; por lo que se decidió realizar la conversión correspondiente con las relaciones entra el tamaño de ambas pantallas y así poder mostrar correctamente la trayectoria que se buscaba.

Una vez solucionado esta complicación, se pudo observar que la raíz del problema era la gran cantidad de sombras y píxeles con un color no muy bien definido debido a la calidad de la WebCam que se usaban por lo que se escogió un dispositivo de mayor calidad para intentar solventar este error y que a la vez fuera de una calidad y rapidez suficiente para poder tratar las imágenes a su debido tiempo.

Con una nueva cámara y unas comparaciones más restrictivas en las funciones de la clase Píxel se consiguió seleccionar de forma mucho más precisa del punto guía (centro) y así poder descartar todos aquellos píxeles que no eran útiles y que tantas imprecisiones habían causado.

- **Selección del objeto adecuado de acuerdo al patrón**

En este caso, se tuvo problemas para determinar cuál era el objeto que se tenía que seleccionar para determinar sus movimientos. En el entorno del usuario pueden encontrarse múltiples objetos del color que hemos de terminado como patrón, por lo que teníamos que filtrar de alguna manera aquellos que no fueran adecuados para el objetivo.

Para solucionarlo, se creado una nueva *clase Objeto* cuyos atributos son el área que ocupa y los puntos característicos de cada uno: superior, inferior y centro.

Todos los objetos candidatos (del color del patrón) que aparecen en la pantalla son guardados en una lista de objetos de la clase anterior para poder determinar su tamaño. Entonces, para sólo estudiar la trayectoria del objeto de interés, se planteó un filtro adecuado: el tamaño. Se ha determinado una aproximación del área que ocupa el objeto determinado y con ello solo plantearse los movimientos que realice.

- **Determinación con cierto retraso de la trayectoria del objeto seleccionado**

El código del programa es bastante complejo y la ejecución de todas las funciones necesarias para utilizar la aplicación requería un coste elevado de tiempo en comparación con los resultados necesarios.

Se tiene que seguir la trayectoria del objeto prácticamente en tiempo real para que el propio usuario pueda percibir un uso muy preciso de todos los movimientos y acciones que va realizando. El más mínimo retraso repercute en una aplicación poco intuitiva y con posibles resultados inadecuados para su correcto funcionamiento, por lo que este problema se debía de corregir con mucha precisión.

Muchos de los errores solventados con anterioridad, perjudican ahora en cuanto a coste en tiempo de ejecución. Por ejemplo, el uso de una WebCam con mayor resolución tiene la desventaja de tener que analizar una imagen con una cantidad mucho mayor de píxeles. Por otra parte al trabajar con una lista de objetos repercute en la creación y eliminación de cada uno de aquellos que eran innecesarios.

Es por ello que se probó seleccionar una imagen más pequeña para la visualización de la cámara y establecer restricciones más complejas para evitar analizar y crear objetos de todo aquello que realmente no era útil en el entorno. Por lo tanto, cada vez que se estudia la imagen de la WebCam, se elimina de la lista aquellos objetos que no son necesarios. Así, se evita recorrer toda la lista posteriormente para encontrar el objeto de interés, y con ello no se tiene que hacer posibles cálculos innecesarios de otros objetos que no sirven para el desarrollo de la aplicación.

De igual forma, se plantea otro problema en la velocidad de ejecución estaba en la función *timer*, en donde se asignaba como valor en tiempo de rango entre las capturas 2000 milisegundos, y al disminuir este tiempo a 500, mejoramos significativamente la velocidad y afinamos la calidad en el trazado de la trayectoria seguida por el objeto.

- **Implementación o simulación de un clic semejante al del ratón**

Para implementar o simular el efecto de un clic de ratón, inicialmente se pensó que con acercar a la cámara WebCam el objeto bastaría. Esto se interpretaría como que al aumentar el número de píxeles rojos del objeto; sería igual al evento del clic del ratón, con lo que se llevaría a cabo la acción consecuente al mismo. Pero a la hora de aplicar esta teoría, se detectó que no por acercar el objeto a la cámara, aumenta el número de píxeles rojos. Es decir, al aumentar el tamaño del objeto rojo, aumenta la superficie que puede verse afectada por el reflejo de la luz; por la tanto los píxeles rojos podrían ser incluso menores que anteriormente, ya que la superficie del objeto donde se refleja la luz, es interpretada como píxeles blancos.

Por tanto, para solucionar el inconveniente el objeto rojo a utilizar debía ser pequeño, y poco reflectante, de manera que podamos reducir el efecto adverso del reflejo de la luz sobre el objeto. De la misma manera el clic del ratón vendría determinado por dos cambios bruscos del número de píxeles rojos del objeto en ocasiones consecutivas dentro de un intervalo de tiempo limitado.

Y tras estos cambio surge un nuevo inconveniente, el pasar de muchos píxeles rojos a pocos, podría ser un clic, pero a su vez podría ser que el usuario ha sacado el objeto del marco de detección de la cámara, pero como obligamos a que deben ser dos clics en un rango de tiempo determinado, se tendría resuelto también este problema; ya que ese clic se interpretaría como ruido y no seria considerado.

Parte 3. RESULTADOS

3.1 Conclusiones

Es más que evidente la importancia del uso de tratamiento de imágenes para la vida cotidiana. Actualmente, hay muchísimas aplicaciones realizadas a través del tratamiento de imágenes para realizar tareas tanto lucrativas y recreativas (videojuegos), como útiles (tratamiento de enfermedades) o informativas e investigación (obtener imágenes del planeta Marte).

En la mayoría de estas aplicaciones hay que tener en cuenta que muchas veces pueden ocurrir errores que planteen problemas en los resultados de la ejecución de la aplicación. El tratamiento de imágenes no se puede realizar siempre de manera exacta porque hay múltiples factores que pueden provocar resultados inexactos. Por ejemplo, entre estos factores se encuentran:

1. La propia luz del entorno que puede complicar gravemente la detección de ciertos patrones de la imagen.

2. La rapidez del tratamiento pues cuanto más rápido se trate una imagen, más imágenes se pueden procesar a la largo del tiempo, pero los resultados pueden ser erróneos y viceversa.

3. La determinación correcta del patrón que puede ser más exacta cuantas más restricciones se implementen pero que conllevará a una ejecución más lenta.

4. La calidad de la imagen puede conllevar a resultados más correctos pero el procesamiento de píxeles es mucho mayor.

Poco a poco, la revolución tecnológica mundial nos hace progresivamente un mundo en el que realizar las tareas más complicadas se ha hecho cada vez más sencillo gracias al uso de los programas informáticos. Unido a esto, se encuentra el tratamiento de imágenes que nos proporciona una ayuda fundamental o complementaria a los resultados del uso cotidiano. Esta rama de la informática cada vez está siendo más utilizada por la sociedad y se encuentra en ámbitos tan diferentes como la medicina, la agricultura o la astrología.

Gracias al tratamiento de imágenes se puede llevar a cabo un estudio más exhaustivo del entorno en el que se encuentra la aplicación y así poder hacer un control más adecuado para las condiciones propias de dicho entorno. Por ejemplo, se puede controlar el riego de una plantación según las condiciones físicas y climatológicas en las que se encuentre un cultivo o podemos determinar con exactitud el comportamiento de unos ciertos patrones que caractericen una aplicación.

3.2 Cumplimiento de los objetivos propuestos

Breve introducción

El objetivo inicial del proyecto consistía en detectar la trayectoria de un objeto determinado para poder simular el movimiento del ratón por el escritorio de un ordenador convencional y detectar los clics correspondientes para seleccionar o ejecutar algún elemento presente en la pantalla.

Para el cumplimiento de este objetivo principal, estructuramos la implementación de nuestra aplicación para conseguir llegar al resultado deseado:

1. Como primer objetivo general debíamos implementar la detección del movimiento de un patrón determinado a través de propiedades prácticamente únicas que lo caracterizasen.

2. Como segundo objetivo general teníamos que conseguir detectar cada clic que hiciera el usuario con el patrón sobre el escritorio y así simular el clic de un ratón sobre la pantalla en unas ciertas coordenadas correspondientes a la situación en la que se encontraba el patrón.

3. Como tercer y último objetivo general debíamos hacer llamadas al sistema correspondientes al clic del ratón sobre unas coordenadas concretas y que el propio puntero correspondiente al ratón se moviera a la par que nuestro objeto patrón.

Objetivos cumplidos

Se ha conseguido cumplir los dos primeros objetivos principales y para representar el tercer objetivo hemos desarrollado tres aplicaciones prácticas para simular el uso de un ratón convencional.

La aplicación es capaz de centrarse correctamente en la trayectoria de un objeto rojo y mostrar gráficamente sobre qué parte del formulario se encuentra el usuario en todo momento. Además es capaz de filtrar otros objetos rojos de nuestro entorno que pueda detectar la WebCam y que realmente no sean útiles para la ejecución adecuada de la aplicación.

También en este proyecto desarrolla la detección del clic del objeto sobre la pantalla de la manera más correcta posible ya que al mover al objeto para simular un clic, la trayectoria del objeto cambia de una forma muy significativa.

Finalmente, se implementan tres aplicaciones del uso práctico que tiene el tratamiento conseguido de las imágenes.

- Para la representación de la detección correcta de la trayectoria se implementa un programa capaz de plasmar dicha trayectoria mediante el dibujo de líneas correspondientes a cada punto por el que pasamos por encima del formulario con nuestro objeto patrón. La idea principal de este programa es poder simular la escritura en una pizarra para que posteriormente una aplicación OCR fuera capaz de detectar lo que hayamos escrito a través de una imagen .JPG que corresponde al formulario sobre el que hemos "escrito" con anterioridad. Además, al igual que una pizarra, también podemos limpiar la pantalla para volver a comenzar la escritura desde el principio.

- Para simular el clic del ratón se han desarrollado dos aplicaciones que simbolizan el uso de un escritorio y el uso de un teléfono convencional. En la primera es posible el movimiento de una carpeta en un escritorio y en la segunda es posible pulsar las teclas comunes de un teléfono cotidiano (las correspondientes a los números en el rango del 0 al 9 y las propias que simulan Llamar y Colgar para la representación gráfica de una llamada al número de teléfono seleccionado por el propio usuario).

3.3 Trabajo futuro

Podrían mejorarse los resultados del tratamiento de imágenes con técnicas de Inteligencia Artificial, como son las Redes Neuronales, y los Algoritmos Genéticos.

También aplicar técnicas para el reconocimiento de patrones, El punto esencial del reconocimiento de patrones es la clasificación: se puede clasificar una señal dependiendo de sus características. Señales, características y clases pueden ser de cualquiera forma, por ejemplo se puede clasificar imágenes digitales de letras en las clases «A» a «Z»

Para la clasificación se puede usar un conjunto de aprendizaje, del cual ya se conoce la clasificación de la información a priori y se usa para entrenar al sistema, siendo la estrategia resultante conocida como aprendizaje supervisado. El aprendizaje puede ser también no supervisado, el sistema no tiene un conjunto para aprender a clasificar la información a priori, sino que se basa en cálculos estadísticos para clasificar los patrones.

Una vez realizada la detección de la trayectoria y el clic con el objeto patrón, se podría ampliar este proyecto para poder llegar al objetivo principal que habíamos planteado desde un principio: manejo de un escritorio virtual. Como ya está implementado la determinación de las coordenadas X e Y sobre las que se hace el clic, habría que investigar para poder enviar información al sistema para conseguir un uso correspondiente al ratón real de nuestros ordenadores.

Para evitar resultados posiblemente erróneos se podría utilizar lógica fuzzy para poder llegar a conclusiones mucho más elaboradas dependiendo del rango de valores que se determinen en función de los resultados deseados.

También se podría hacer un estudio elaborado del comportamiento humano en el uso de la aplicación a través del procesamiento de cada una de las acciones que sean llevadas a cabo. A partir de este estudio se podría hacer un seguimiento más adecuado de los clics realizados con el objeto patrón y así conseguir un uso mucho más sencillo e inteligente.

Para poder identificar la escritura realizada sobre el formulario de la aplicación *representación de la trayectoria del objeto*, se podría utilizar un software de **reconocimiento óptico de caracteres**, abreviado habitualmente como **OCR** (*Optical character recognition*), que extrae de una imagen los caracteres que componen un texto para almacenarlos en un formato con el cual puedan interactuar programas de edición de texto. De manera que se capaz de determinar correctamente las formas que se encuentran en la imagen del formulario y así poder escribir en el ordenador a mano alzada utilizando un objeto adecuado para la escritura.

Parte 4. ANEXOS

Anexo 1

Java Media Framework

Introducción

La librería Java Media Framework (JMF) es una extensión de Java para poder realizar tareas multimedia con más facilidad en este lenguaje de programación. Esta especializada en las tareas multimedia que requieren un tiempo corto de respuesta o incluso requieren esta respuesta en tiempo real.
Para este proyecto se ha utilizado la versión JMF 2 que permite capturar y almacenar datos multimedia y controlar el tipo de procesamiento durante la retransmisión entre otros. Los objetivos del API JMF 2 con respecto a la JMF 1.0 que nos hacen decantarnos por esta versión son:

- Programar con facilidad.
- Permitir la captura de datos multimedia.
- Permitir el flujo multimedia y las aplicaciones con conferencia en Java.
- Permitir a los desarrolladores avanzados y proveedores implementar soluciones personalizadas basándose en los API existentes e integrar con facilidad características nuevas con el código actual.
- Proveer el acceso a los datos multimedia “raw” (Son datos multimedia “en crudo” sin utilizar ningún tipo de artefacto para tener que tratarlos)
- Permitir el desarrollo de codecs, efectos de procesamiento, multiplexores, demultiplexores y “renderers” personalizados.
- Mantener la compatibilidad con JMF 1.0.

Por otra parte, es necesario instalar JMF como un paquete adicional ya que no se incluye en la JDK o JRE de la maquina virtual de java.

Características

Sus características principales son:

- Gran estabilidad, al estar funcionando sobre la máquina virtual de java.
- Sencillez, ya que permite, usando unos pocos comandos, realizar complejas tareas multimedia.
- Potencia, permitiendo la manipulación de elementos multimedia de vídeo y audio locales (procedentes de la misma máquina en la que se ejecuta el programa), así como la retransmisión en tiempo real de vídeo y audio a través de la red mediante el protocolo RTP.

Licencia:

El código de JMF esta accesible bajo el programa Sun Community Licensing (SCSL), que puede resumirse según sea o no para uso comercial.

Para uso no comercial:

SCSL permite la distribución de código fuente para uso no comercial, evaluación y/ o investigación.

Para uso comercial:

Si algún cliente quiere realizar la licencia de código fuente JMF para un producto comercial, deberá realizar un "Commercial Use Attachment" con la compañía Sun, entre otros.

Formatos:

JMF puede soportar y utilizar una serie de formatos multimedia indicados a continuación:

AIFF (.aiff)

- 8-bit mono/stereo linear
- 16-bit mono/stereo linear
- G711 (U-law)
- A-law (Solo para decodificación y presentación)
- IMA 4 ADPCM

AVI (.avi)

- Audio: 8-bit mono/stereo linear
- Audio: 16-bit mono/stereo linear
- Audio: DVI ADPCM comprimido
- Audio: G.711 (U-law)
- Audio: A-law (Solo para decodificación y presentación
- Audio: GSM mono
- Audio: ACM (Solo para el paquete optimizado para Windows)
- Video: Cinepak (Solo para decodificación y presentación en paquetes distintos del optimizado para Linux/Solaris)
- Video: MJPEG (422) (Solo para decodificación y presentación en paquetes optimizados)
- Video: RGB
- Video: YUV
- Video: VCM (Solo para el paquete optimizado para Windows)

GSM (.gsm)

- GSM mono audio

HotMedia (.mvr)

- IBM HotMedia (Solo para decodificación y presentación)

MIDI (.mid)

- Type 1 & 2 MIDI (Solo para decodificación y presentación en paquetes optimizados)

MPEG-1 Video (.mpg)

- Multiplexed System stream (Solo para decodificación y presentación en paquetes optimizados)
- Video-only stream (Solo para decodificación y presentación en paquetes optimizados)

MPEG Layer II Audio (.mp2)

- MPEG layer 1,2 audio (La codificación solo esta disponible en paquetes optimizados)

QuickTime (.mov)

- Audio: 8 bits mono/stereo linear
- Audio: 16 bits mono/stereo linear
- Audio: G.711 (U-law)
- Audio: A-law (Solo para decodificación y presentación)
- Audio: GSM mono
- Audio: IMA4 ADPCM
- Video: Cinepak (La codificación solo esta disponible en paquetes optimizados)
- Video: H.261 (Solo para decodificación y presentación en paquetes optimizados)
- Video: H.263 (La codificación solo esta disponible en paquetes optimizados
- Video: JPEG (420, 422, 444) (La codificación solo esta disponible en paquetes optimizados)
- Video: RGB

Sun Audio (.au)

- 8 bits mono/stereo linear

- 16 bits mono/stereo linear
- G.711 (U-law)
- A-law (Solo para decodificación y presentación)

Wave (.wav)

- 8-bit mono/stereo linear
- 16-bit mono/stereo linear
- G.711 (U-law)
- A-law (Solo para decodificación y presentación)
- GSM mono
- DVI ADPCM
- MS ADPCM (Solo para decodificación y presentación)
- ACM (Solo para el paquete optimizado para Windows)

También puede enviar o recibir los siguientes formatos RTP:

- Audio: G.711 (U-law) 8 kHz
- Audio: GSM mono
- Audio: G.723 mono (Solo para decodificación y presentación)
- Audio: 4-bit mono DVI 8 kHz
- Audio: 4-bit mono DVI 11.025 kHz
- Audio: 4-bit mono DVI 22.05 kHz
- Audio: MPEG Layer I, II
- Video: JPEG (420, 422, 444)
- Video: H.261(Solo para decodificación y presentación)
- Video: H.263
- Video: MPEG-I

JMF admite los siguientes dispositivos de captura:

- JavaSound (16-bit, 44100, 22050, 11025Hz, 8000Hz linear)
- SunVideo (Solo para el paquete optimizado de Solaris)
- SunVideoPlus (Solo para el paquete optimizado de Solaris)
- VFW (Solo para el paquete optimizado de Windows)

- Intel Create & Share (Solo para el paquete optimizado de Windows bajo plataformas Win9x)
- Diamond Supra Video Kit; Share (Solo para el paquete optimizado de Windows bajo plataformas Win98)
- QuickCam VC (camera) (Solo para el paquete optimizado de Windows bajo plataformas WinNT)
- e-cam (camera) (Solo para el paquete optimizado de Windows bajo plataformas WinNT, Win9x)
- Winnow Videum (Solo para el paquete optimizado de Windows bajo plataformas WinNT, Win9x)
- Creative Web Cam II (Solo para el paquete optimizado de Windows bajo plataformas Win9x)
- Miro Video DC30 (Solo para el paquete optimizado de Windows bajo plataformas Win9x)
- Iomega Buz (Solo para el paquete optimizado de Windows bajo plataformas Win9x)
- QuickCam Home USB (Camera) (Solo para el paquete optimizado de Windows bajo plataformas Win98)
- Smart Video Recorder III (Solo para el paquete optimizado de Windows bajo plataformas Win9x)

Descripción de entorno multimedia

Cualquier dato que cambia significativamente con respecto al tiempo puede ser caracterizado como multimedia basada en el tiempo. Clips de audio, secuencias midi, clips de películas y animaciones son una forma común de multimedia basada en el tiempo. Así mismo, los datos multimedia pueden obtenerse de diferentes fuentes como micrófonos, cámaras o archivos de red. JMF se encarga de gestionar los dispositivos de entrada y salida multimedia y el pre y post-procesamiento de la información multimedia basada en el tiempo.

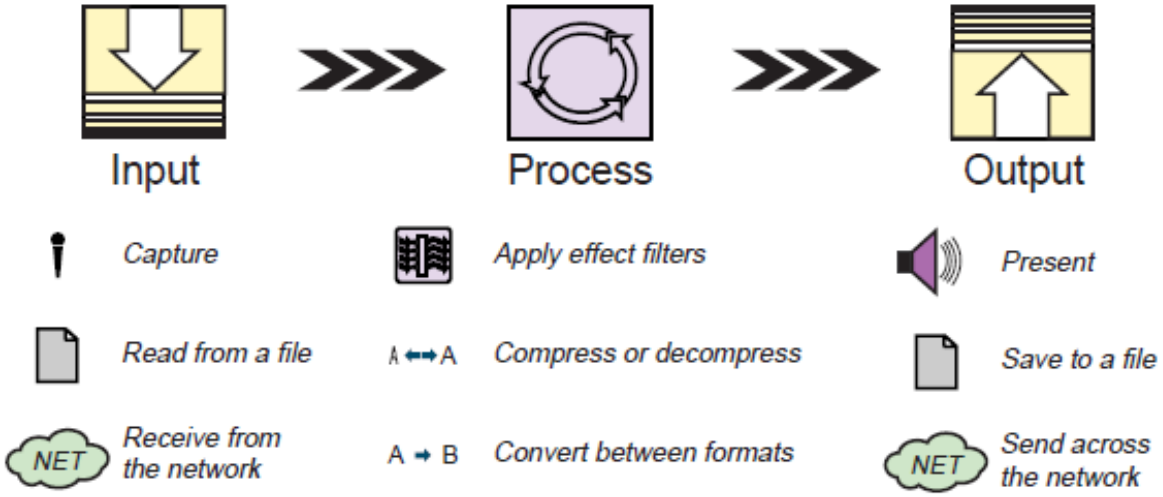


Figura 1: Esquema de funcionamiento de JMF.

Streaming Media

Una característica clave de la multimedia basada en el tiempo es que requiere un procesamiento y envió en tiempo. Una vez que el flujo de datos multimedia comienza, deben realizarse las tareas con respecto a los datos en un margen de tiempo bien delimitado, para poder recibir y representar la información. Por esta razón, se suele referir la multimedia basada en el tiempo como "Streaming media".

Por ejemplo, cuando se reproduce una película, si no se puede distribuir la información multimedia lo suficientemente rápido, se pueden producir pausas y retrasos en la retransmisión. Por otra parte, si los datos no se pueden recibir y procesar lo suficientemente rápidos, la película puede realizar saltos como si la información se hubiese perdido o como si se hubiese recortado algunos "Frames" de forma intencionada para intentar mantener una velocidad de reproducción adecuada.

Tipo de contenido

El formato en el que se almacena la información multimedia se refiere como el tipo de contenido. QuickTime, MPEG, y WAV son ejemplo de tipos de contenido. El tipo de contenido es en esencia un sinónimo de tipo de archivo. Se usa el tipo de contenido porque los datos multimedia se pueden adquirir de otras fuentes que no sean archivos locales.

Flujos Multimedia

Un flujo multimedia son los datos multimedia obtenidos de un archivo local, recogidos por la red o capturados de una cámara o micrófono. Los flujos multimedia suelen contener múltiples canales de información llamados pistas o "tracks". Por ejemplo, u archivo Quicktime pueden contener tanto una pista de audio como una de video. Los flujos multimedia que contienen múltiples pistas suelen referirse como flujos multimedia multiplexados o complejos. Demultiplexar es el proceso de extraer las pistas individualmente de un flujo multimedia complejo.

Un tipo de pista identifica el tipo de datos que contiene, como audio o video. El formato de una pista define como están estructurados los datos de la pista.

Un flujo multimedia puede definirse por su localización y el protocolo que se usa para acceder a el. Por ejemplo, una URL puede usarse para describir la localización de un archivo QuickTime en un sistema local o remoto. Si el archivo es local, puede accederse a el mediante un protocolo FILE. Por otra lado, si esta en un servidor web, se puede acceder al archivo mediante el protocolo HTTP. Un localizador multimedia aporta una forma de identificar la localización de un flujo multimedia cuando no se puede usar una URL.

Los flujos multimedia pueden organizarse de acuerdo a como se distribuyen los datos:

- Pull – La transferencia se inicia y controla por el cliente. Por ejemplo, HTTP y FILE son protocolos Pull.
- Push – El servidor inicia la trasferencia de datos y controla el flujo de datos. Por ejemplo, RTP es un protocolo push que se usa para el flujo multimedia.
-

Formatos multimedia comunes

Las siguientes tablas identifican algunas de las características de formatos multimedia comunes. Cuando se selecciona un formato, es importante tener en cuenta las características del mismo, el objetivo del entorno y las expectativas del cliente al que va destinado. Por ejemplo, si distribuyes contenido multimedia a través de la web, necesitas prestar una atención especial a los requisitos del ancho de banda.

La columna de requisitos de CPU indica el poder de procesamiento necesario para obtener una representación óptima del formato especificado. La columna de ancho de banda indica la velocidad de retransmisión necesaria para enviar o recibir datos para una representación óptima.

Formato	Tipo de contenido	Calidad	Requisitos de CPU	Requisitos de ancho de banda
Cinepak	AVI QuickTime	Media	Bajos	Altos
MPEG-1	MPEG	Alta	Altos	Altos
H.261	AVI RTP	Baja	Medio	Medio
H.263	QuickTime AVI RTP	Media	Medio	Medio
JPEG	QuickTime AVI RTP	Alta	Altos	Altos
Indeo	QuickTime AVI	Media	Medio	Medio

Tabla 1: Formatos de video comunes

Formato	Tipo de contenido	Calidad	Requisitos de CPU	Requisitos de ancho de banda
PCM	AVI QuickTime WAV	Alta	Bajos	Altos
Mu-law	AVI QuickTime WAV RTP	Baja	Bajos	Altos

ADPCM (DVI,IMa4)	AVI QuickTime WAV RTP	Media	Medios	Medios
MPEG-1	MPEG	Alta	Altos	Altos
MPEG Layer 3	MPEG	Alta	Altos	Medio
GSM	WAV RTP	Baja	Bajos	Bajos
G.723.1	WAV RTP	Media	Medio	Medio

Tabla 2: Formatos de audio comunes

Algunos formatos se diseñan para una aplicación en particular unos requisitos pensados de antemano. Los formatos de alta calidad y de un gran ancho de banda suelen usarse a través de dispositivos de almacenamiento o para almacenamiento local de aplicaciones. H.261 y H263 se suelen usar generalmente para aplicaciones de video conferencia y están optimizados para videos con poco tratamiento.

Representación multimedia

La mayor parte de los datos multimedia basados en el tiempo son datos de audio o video que pueden representarse a través de dispositivos de salida como altavoces o monitores. Estos dispositivos suelen ser el destino más común para la salida de datos multimedia. Los flujos multimedia pueden enviarse a otras aplicaciones – por ejemplo, guardarlos en un archivo o retransmitirlos por la red. Un destino de salida para los datos multimedia es conocido a veces como un sumidero de datos (“Data sink”).

Controles de representación

Mientras un flujo multimedia puede ser representado, los controles de representación VCR-style suelen aportar al usuario la posibilidad de controlar la reproducción. Por ejemplo, un panel de control para un reproductor de películas suele ofrecer botones para parar, comenzar, avanzar o rebobinar la película.

Latencia

En muchos casos, especialmente cuando se representa un flujo multimedia que se obtiene de la red, la representación del flujo multimedia no puede comenzar inmediatamente. El tiempo que requiere antes de que la representación pueda comenzar se conoce como latencia de inicio ("start latency"). Los usuarios pueden experimentar un retraso entre el tiempo que ellos hacen clic en el botón de inicio, y el tiempo en que la reproducción comienza.

Calidad de la representación

La calidad de la representación de un flujo multimedia depende de muchos factores, entre los que se incluyen:

- El esquema de compresión usado
- La capacidad de procesamiento de reproducción del sistema
- El ancho de banda disponible

Tradicionalmente, cuanto mayor calidad, mayor es el tamaño del fichero y mayor el poder de procesamiento y el ancho de banda requerido. El ancho de banda normalmente se representa como el numero de bits que se transmiten en cierto periodo de tiempo- el ratio de bits (Bits rate).

Para alcanzar una representación con una alta calidad de video, el numero de "frames" mostrados en cada periodo de tiempo (el ratio de frames, "frame rate") debe ser los más alto posible. Normalmente, se consideran las películas con un "frame rate" de 30 frames por segundo como indistinguibles de las emisiones de televisión.

Procesamiento multimedia

En la mayoría de casos, los datos en un flujo multimedia se manipulan antes de que se muestren al usuario. Normalmente, se suceden una serie de operaciones de procesamiento antes de la representación:

1. Si el flujo esta multiplexado, se extraen las pistas de forma individual.
2. Si las pistas individuales están comprimidas, se decodifican.
3. Si es necesario, las pistas se convierten a un formato diferente.
4. Se aplican los filtros de efectos a las pistas decodificadas. (Si se quiere)

Las pistas se distribuyen al dispositivo de salida apropiado. Si el flujo multimedia puede almacenarse en vez de mostrarse en un dispositivo de salida, las etapas de procesamiento pueden variar ligeramente. Por ejemplo, si quiere capturar audio y video de una cámara de video, procesar los datos y guardarlos en un archivo:

1. Las pistas de audio y video se capturan.
2. Se aplican los filtros a las pistas puras (Si se quiere).
3. Las pistas individuales se codifican.

4. Las pistas comprimidas se multiplexaran en un flujo multimedia único.
5. El flujo de datos multiplexado se guarda entonces en un archivo.

Multiplexores y demultiplexores

Un demultiplexor extrae pistas individuales de datos multimedia de un flujo multimedia multiplexado. Un multiplexor realiza la función opuesta, toma las pistas individuales de datos multimedia y las une en un único flujo multimedia multiplexado.

Codecs

Un codec realiza una compresión y una descompresión de un dato multimedia. Cuando una pista se codifica, se convierte a un formato comprimido adecuado para almacenamiento o retransmisión; cuando se decodifica se convierte a un formato no-comprimido adecuado para representación.

Filtros de efectos

Un filtro de efecto modifica el dato de la pista de alguna forma, normalmente para crear un efecto especial como un difuminado o un eco.

Los filtros de efectos se clasifican en efectos de pre-procesamiento o de post-procesamiento, dependiendo de si se aplican antes o después de que el codec procese la pista. Normalmente los filtros de efecto se aplican a datos descomprimidos.

Renderizadores

Un renderizador en una abstracción de un dispositivo de representación. Para el audio, el dispositivo de representación suele ser una tarjeta de audio que enviá sonido a los altavoces. Para el video, la representación del dispositivo suele ser el monitor del ordenador.

Composición

Ciertos dispositivos especializados soportan composición. Componer multimedia basada en el tiempo es el proceso de combinar varias pistas de datos en un único medio de representación. Por ejemplo, superponer texto en una representación de video es una de las formas más comunes de composición. Un dispositivo que realiza composición puede se abstraído como un renderizador que puede recibir múltiples pistas de datos de entrada.

Captura de medios

Se puede capturar multimedia basada en el tiempo de una fuente en vivo para procesarla y reproducirla. Por ejemplo, el audio puede capturarse de un micrófono o una tarjeta de captura de video puede usarse para obtener video de una cámara. Capturar puede pensarse como la fase de entrada de un modelo de procesamiento multimedia estándar.

Un dispositivo de captura puede enviar múltiples flujos multimedia. Por ejemplo, una cámara de video puede enviar tanto video como audio. Estos flujos pueden ser capturados y manipulados de forma separada o combinados en único flujo multiplexado que contiene tanto pistas de audio como pistas de video.

Dispositivos de captura

Para capturar multimedia basada en el tiempo necesitas hardware especializado – por ejemplo, para capturar audio de una fuente en vivo, necesitas un micrófono y una tarjeta de audio apropiada. De igual forma, capturar emisiones de televisión requiere una sintonizadora de televisión y una tarjeta de captura de video adecuada. La mayoría de sistemas suministran mecanismos de cola para encontrar los dispositivos de captura disponibles.

Los dispositivos de captura pueden caracterizarse por fuentes de push o fuentes de pull. Por ejemplo, una cámara fija es una fuente pull – el usuario controla cuando capturar la imagen. Un micrófono es una fuente push – el entorno continuamente proporciona un flujo de audio.

El formato de un flujo multimedia capturado depende de la transformación realizada por el dispositivo de captura. Algunos dispositivos realizan muy poco procesamiento y envían datos puros sin comprimir. Otros dispositivos de captura pueden enviar los datos en un formato comprimido.

Controles de captura

Los controles a veces permiten al usuario dirigir el proceso de captura. Por ejemplo, un panel de control de captura puede permitir al usuario especificar el ratio de datos y el tipo de codificación para el flujo de captura y el inicio y fin del proceso de captura.

Descripción a alto nivel de la arquitectura JMF

Dispositivos como un DVD y un reproductor de DVD permiten un modelo familiar para almacenar, procesar y representar multimedia basada en el tiempo. Cuando reproduces una película usando un reproductor de DVD, proporcionan un flujo multimedia al reproductor insertando el DVD. El reproductor lee e interpreta los datos del DVD y manda la señal apropiada a la televisión y a los altavoces.

JMF usa el mismo modelo básico. Una fuente de datos (“data source”) encapsula el flujo multimedia como un DVD y un reproductor (“player”) suministra procesamiento y mecanismos de control similares al reproductor de DVD. Reproducir y capturar audio y video con JMF requieren los dispositivos de entrada y salida apropiados como micrófonos, cámara, altavoces y monitores.

Las fuentes de datos y reproductores son partes integradas del API de JMF para manejar la captura, representación y procesamiento de multimedia basada en el tiempo. JMF también proporciona una API a bajo nivel que permite la integración sin fisuras de componentes y extensiones de procesamiento personalizados. Esta distinción de capas proporciona a los desarrolladores de Java un API con facilidades para incorporar multimedia basada en el tiempo en programas Java manteniendo la flexibilidad requerida para soportar aplicaciones multimedia avanzadas y futuras tecnologías multimedia.

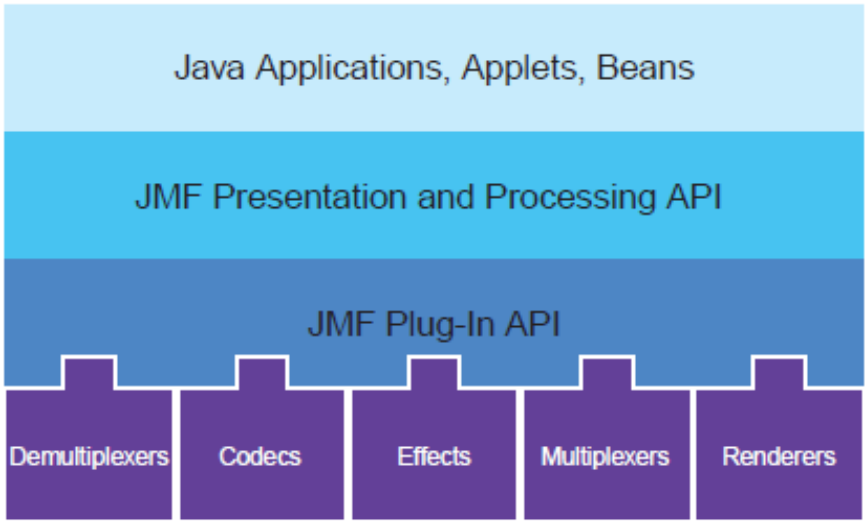


Figura 2: Modelo de capas de JMF

Modelo de tiempo

JMF mantiene el tiempo con una precisión de nanosegundos. Un punto particular en el tiempo se suele representar como un objeto *Time*, pensando que algunas clases soportan las especificación del tiempo en nanosegundos.

Las clases que permiten el modelo de tiempo JMF implementan *Clock* para mantener constancia del tiempo para un flujo multimedia en particular. La interfaz *Clock* define el tiempo base y las operaciones de sincronización que se necesitan para controlar la representación de datos multimedia.

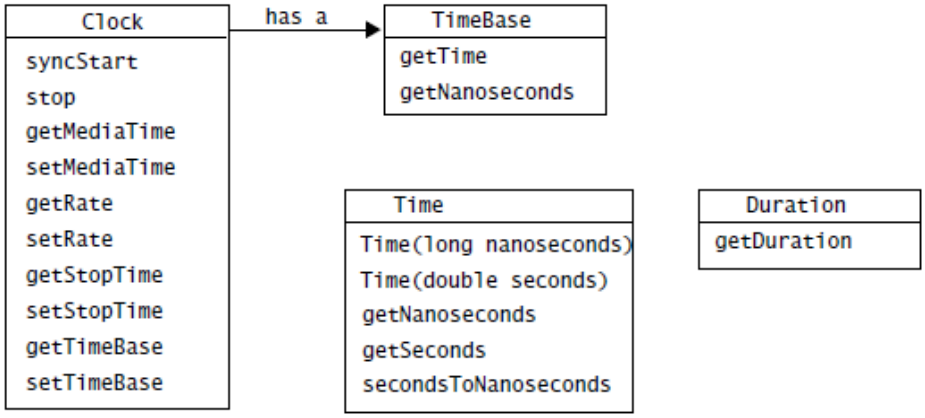


Figura 3: Diagrama de las clases de Clock,TimeBase,Time y Duration

Un *Clock* usa a *TimeBase* para mantener constancia de el paso del tiempo mientras el flujo multimedia se esta representando. Un *TimeBase* proporciona constantemente una fuente de ticks temporales, como un péndulo oscilando en un reloj. La única información que un *TimeBase* proporciona es su tiempo actual, al que se le refiere como el tiempo base. El tiempo base no se puede parar o resetear. El tiempo base a menudo esta basado en el reloj del sistema.

El tiempo multimedia de un objeto *Clock* representa la posición actual en el flujo multimedia – el principio de el flujo es el tiempo cero, el final del flujo es el máximo tiempo multimedia del flujo. La duración del flujo multimedia es el tiempo que ocurre desde el principio al final – la longitud de tiempo que toma representar el flujo multimedia. (Los objetos multimedia implementan la interfaz *Duration* si pueden informar de la duración del flujo multimedia).

Para mantener constancia del tiempo multimedia actual, un *Clock* usa:

- El inicio del tiempo base – El tiempo que da *TimeBase* cuando la representación comienza.
- El inicio del tiempo multimedia – La posición en el flujo multimedia donde la representación comienza.
- El ratio de reproducción – Lo rápido que esta corriendo el *Clock* en relación a su *TimeBase*.

Cuando la representación comienza, el tiempo multimedia se mapea al tiempo base y el avanza del tiempo base se usa como medida del paso del tiempo. Durante la representación, el tiempo multimedia actual se calcula usando la siguiente formula:

```
MediaTime = MediaStartTime + Rate(TimeBaseTime - TimeBaseStartTime)
```

Cuando se para la representación, el tiempo multimedia se para, pero el tiempo base continua avanzando. Si la representación se reanuda, el tiempo multimedia se remapea a el tiempo base actual.

Directores (Managers)

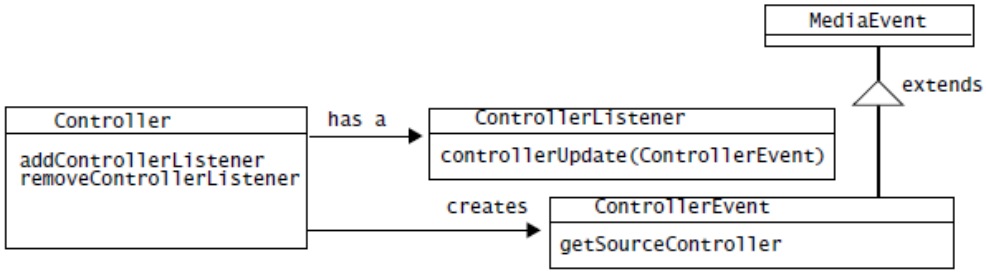
La API JMF consiste principalmente en interfaces que definen el comportamiento y la interacción de objetos usados para capturar, procesar y representar el tiempo base multimedia. Las implementaciones de estas interfaces operan dentro de la estructura del entorno de trabajo. Usando objetos intermediarios llamados *managers*, JMF facilita la integración de nuevas implementaciones de interfaces clave que pueden usarse de forma parecida con clases existentes.

JMF usa cuatro managers:

- *Manager* – Dirige la construccion de *Players*, *Processors*, *DataSources* y *DataSinks*.
- *PackageManager* – Mantiene un registro con los paquetes que contiene clases JMF, como *Players*, *Processors*, *DataSources* y *DataSinks* personalizadas.
- *CaptureDeviceManager* – Mantiene un registro con los dispositivos de captura disponibles.
- *PlugInManager* – Mantiene un registro de los plug-in JMF de componentes de procesamiento, como *Multiplexers*, *Demultiplexers*, *Codecs*, *Effects* y *Renderers*.

Para escribir programas basados en JMF, se necesita usar el metodo *Manager create* para construir *Players*, *Processors*, *DataSources* y *DataSinks* para las aplicaciones. Si se captura información multimedia de un dispositivo de captura, se debe usar *CatureDeviceManager* para encontrar que dispositivos están disponibles y para acceder a información sobre ellos. Si interesa controlar el proceso que es esta realizando a la información, se debe usar Plug-InManager para determinar que plug-ins están registrados.

Modelo de eventos



JMF usa un mecanismo de reporte de eventos estructurados para mantener los programas basados en JMF informados del estado actual del sistema multimedia y permitiendo a los programas basados en JMF responder a las condiciones de error multimedia como a falta de datos. Siempre que un objeto JMF necesita avisar de su actual condición, avisa a MediaEvent, que es una subclase para identificar algunos tipos particulares de eventos.

Figura 4: Diagrama de clases del modelo de eventos de JMF

El objeto *Controller* (Como *Players* y *Processors*) y ciertos objetos *Control* como *GainControl* mandan eventos multimedia.

Modelo de datos

Los reproductores multimedia JMF normalmente usan *DataSources* para dirigir la transferencia de contenido multimedia. Un DataSource encapsula tanto la localización de multimedia como el protocolo y software usado para su retransmisión. Una vez obtenido, la fuente no puede reutilizarse para retransmitir otros datos multimedia.

Un *DataSource* se identifica por un JMF *MediaLocator* o por una *URL*.Un MediaLocator es similar a un URL y puede ser construido a partir de un URL, pero se puede construir incluso si el controlador del protocolo no esta instalado en el sistema.

Un DataSource dirige un conjunto de objetos SourceStream. Una fuente de datos estándar usa un array de bytes como unidad de transferencia. Una fuente de datos de buffer usa un objeto Buffer como una unidad de transferencia. JMF define varios tipos de objetos DataSource:

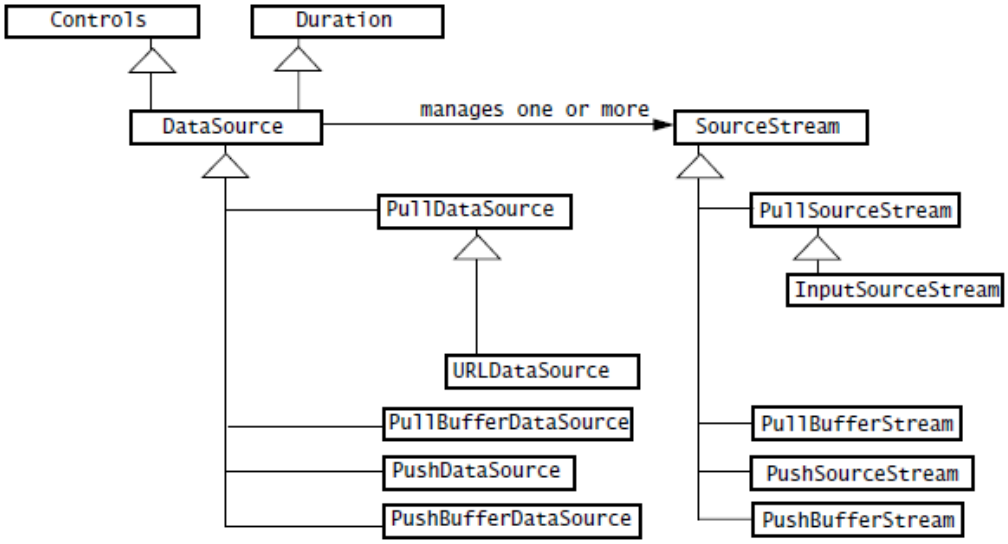


Figura 5: Diagrama de clases del modelo de datos de JMF.

Los datos multimedia se pueden obtener de una variedad de fuentes, como archivos locales o de red. Las fuentes de datos de JMF pueden organizarse según como se inicio la transferencia:

- Fuentes de datos Pull
- Fuentes de datos Push

El formato multimedia exacto de un objeto se representa por un objeto *Format*. El formato en si mismo lleva parámetros no específicos de la codificación o información del tiempo global.

JMF extiende *Format* para definir formatos de video y de audio específicos.

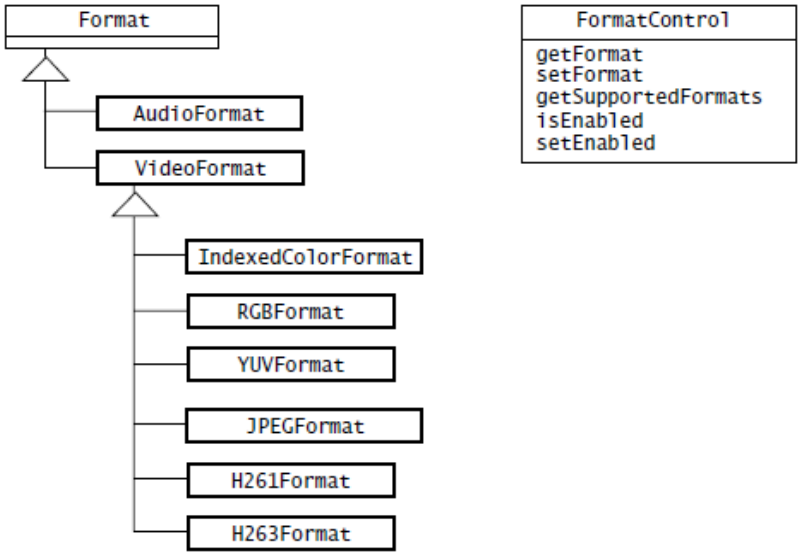


Figura 6: Diagrama de clases que extienden de la clase Format.

Controles

JMF *Control* establece un mecanismo para editar y consultar atributos de un objeto. Un Control suele proporcionar acceso al componente de la interfaz de usuario correspondiente que permite al usuario controlar los atributos de un objeto. Algunos objetos JMF exponen *Controls*, incluyendo objetos *Controller*, objetos *DataSource*, objetos *DataSink* y plug-ins JMF.

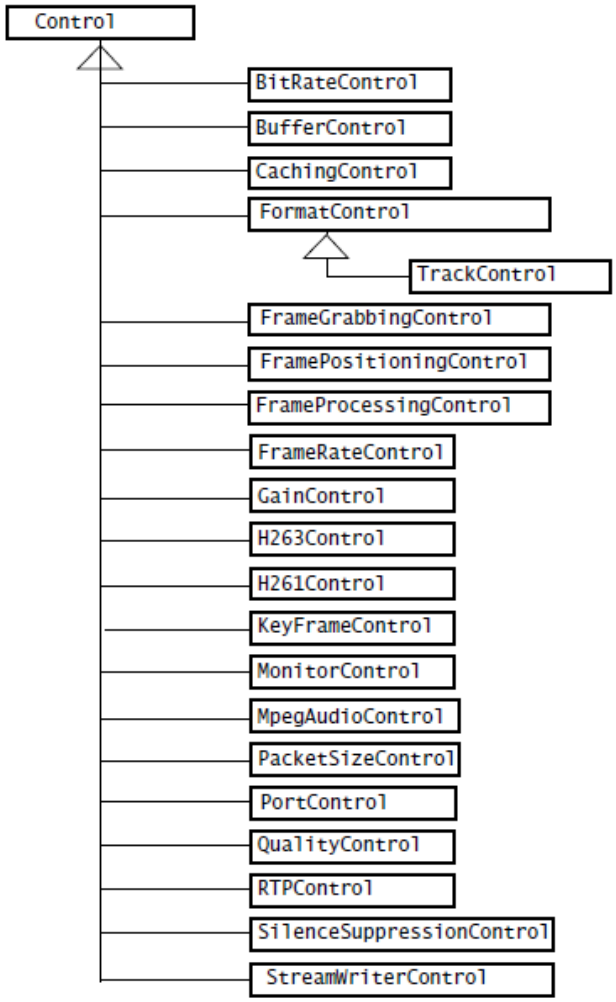


Figura 7: Diagrama de las clases Control.

Componentes de la interfaz de usuario

Un *Control* puede proporcionar acceso a la interfaz de usuario *Component* que expone su comportamiento de control al usuario final. Para obtener el componente de la interfaz de usuario para un *Control* particular, se llama a *getControlComponent*. Este método devuelve un *Component* AWT que se puede añadir al espacio de presentación de un applet o a una aplicación con ventanas.

Presentación

En JMF, el proceso de presentación esta modelado la interfaz *Controller*. *Controller* define el estado básico y el mecanismo de control para un objeto que controla, presenta o captura multimedia basada en el tiempo. Define las fases por las que un controlador multimedia pasa y proporciona un mecanismo para controlar la transición entre estas fases.

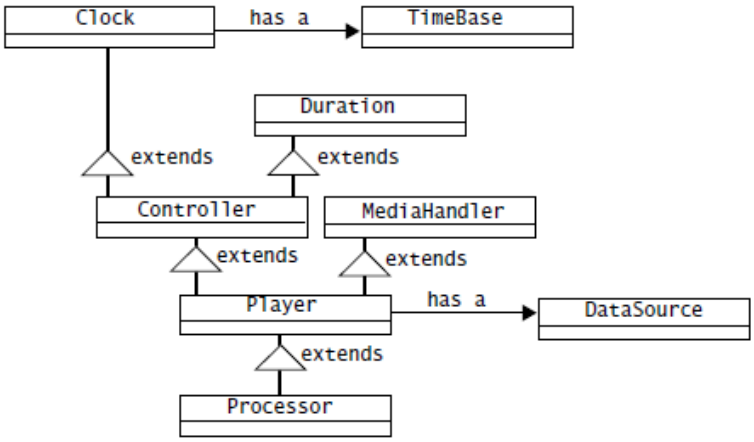


Figura 8: Diagrama de clases de la presentación JMF

Players

Un *Player* procesa un flujo de entrada de datos multimedia y lo renderiza en el momento preciso. Un *DataSource* se usa para enviar el flujo de entrada multimedia a el *Player*. El destino de renderización depende del tipo de elemento multimedia en el que se representa.

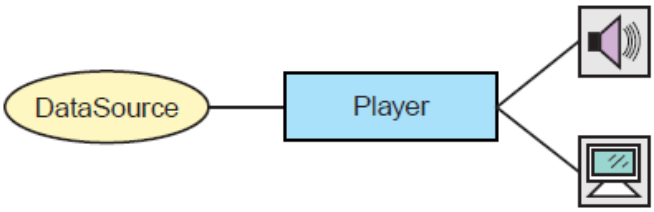


Figura 9: Diagrama de la relación DataSource - Player

Processors

Los *Processors* pueden usarse para representar datos multimedia. Un *Processor* es simplemente un tipo especializado de *Player* que proporciona control sobre que procesamiento se esta realizando en el flujo de entrada multimedia. Un *Processor* soporta todos los mismos controles de representación que un *Player*.

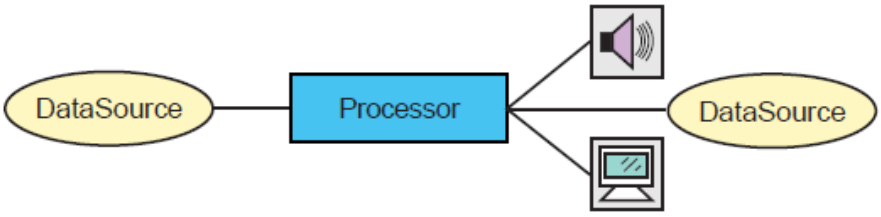


Figura 10: Diagrama de la relación DataSource - Processor

Captura

Un dispositivo de captura multimedia puede actuar como una fuente de envió de datos multimedia. Por ejemplo, un micrófono puede capturar un entrada de audio puro. Este tipo de dispositivos de captura se abstraen como *DataSources*. Cualquier tipo de *DataSource* puede ser usado como una captura *DataSource*: *PushDataSource*, *PushBufferDataSource*, *PullDataSource* o *PullBufferDataSource*.

Almacenamiento de datos multimedia y retransmisión

Un DataSink se usa para leer datos multimedia de un DataSource y renderizar el elemento multimedia en algún destino. En particular un DataSink puede escribir datos a un archivo, escribir datos a través de la red o funcionar como un radiodifusor RTP.

Anexo 2

Licencia JMF

Java Media Framework (JMF) 2.1.1

Licencia de Código Binario

Para poder utilizar el JMF, Sun le obliga a leerse los términos de este acuerdo y los términos de licencia proporcionados de manera suplementaria (En general "el acuerdo") antes de abrir el paquete de este software. Al abrir el paquete, Sun considera que acepta los términos de este acuerdo. Si se utiliza este software de forma electrónica, debería indicarse la aceptación de estos términos accediendo a la Web de Sun y dejando constancia en el acuerdo de la propia Web. Si no se aceptan todos estos términos, se debe devolver inmediatamente el software no utilizado a su lugar de origen o si su acceso fue electrónico, seleccione la opción Rechazar al final del acuerdo.

1. **Licencia de uso.**
   Sun Microsystems, Inc. ("Sun") le concede una licencia no exclusiva e intransferible para el uso interno del software, la documentación y cualquier corrección de errores suministrada por Sun (colectivamente "Software"), por el número de los usuarios y la clase de equipo para el que se va a utilizar.

2. **Restricciones.**

   El software es confidencial y tiene derechos de autor. Ha de tenerse en cuenta que el título de software y todos los derechos de propiedad intelectual son propiedad de Sun y / o sus adheridos a la licencia. Con excepción de lo expresamente autorizado en cualquier término adicional de licencia, no se pueden hacer copias de software, con excepción de una única copia del Software con fines de ser archivado. A menos que la ejecución esté prohibida por la ley, no se puede modificar, descompilar o utilizar técnicas de ingeniería inversa de software. Al ser utilizado, se reconoce que el software no está diseñado o destinado a ser utilizado para el diseño, construcción, operación o mantenimiento de cualquier instalación nuclear. Sun renuncia a cualquier garantía expresa o implícita de adecuación para tales usos. Ningún derecho, título o interés en o para cualquier marca de servicio, logotipo o nombre comercial de Sun o de sus otorgantes de licencias se conceden en virtud del presente acuerdo.

3. **Garantía Limitada**

   Con este acuerdo, Sun le garantiza que durante un período de noventa días, los medios en los que se suministra software (en su caso) estarán libres de defectos en materiales y mano de obra bajo uso normal. Sus recursos exclusivos, además de toda la

responsabilidad de Sun bajo esta garantía limitada le permitirá a Sun reemplazar los medios de suministración.

4. Renuncia de garantía

A menos que se especifique en este acuerdo, todas las condiciones expresadas o implícitas, representaciones y garantías, incluyendo cualquier garantía implícita de comerciabilidad, idoneidad para un propósito particular se consideran legalmente inválidas.

5. Limitación de responsabilidad.

En la medida de todo aquello no prohibido por la propia ley, en ningún caso Sun le informa que ni la propia empresa ni sus proveedores de licencias serán responsables de cualquier pérdida de ingresos, beneficios o datos, o de daños especiales, indirectos, consecuentes, incidentales o punitivos, independientemente de cualquier responsabilidad, que puedan surgir o estén relacionados con el uso o la imposibilidad del uso del software, incluso si Sun es consciente de la posibilidad de dichos daños. Las limitaciones se aplican incluso si la garantía falla en su propósito esencial.

6. Terminación

Este acuerdo estará vigente hasta su vencimiento. Sun le facilita la posibilidad de acabar con el presente acuerdo en cualquier momento destruyendo todas las copias del Software. El presente contrato se resolverá de inmediato y sin previo aviso de Sun, si usted no cumple alguna de las disposiciones del presente acuerdo. Tras la terminación, usted debe destruir todas las copias del Software.

7. Normativa de exportación

Todo el software y los datos técnicos entregados en virtud de este acuerdo están sujetos a las leyes de control de la exportación de EE.UU. y pueden estar sujetos a normas de importación o exportación en otros países. Sun le obliga a aceptar el compromiso de cumplir estrictamente con todas las leyes y reglamentos, y reconocer que usted tiene la responsabilidad de obtener tales licencias de exportación, reexportación o importación que puedan ser necesarios después de la entrega del software.

8. Derechos limitados del gobierno norteamericano

Si el software es adquirido por o en nombre del Gobierno de los EE.UU. o por un contratista principal y subcontratistas (de cualquier nivel) del Gobierno de los EE.UU., entonces los derechos del Gobierno del software y la documentación adjunta será aplicable sólo como se establece en el presente acuerdo, conforme con 48 CFR 227.7202-4 (para el Departamento de Defensa (DOD) adquisiciones) y con 48 CFR 2.101 y 12.212 (para adquisiciones ajenas al DOD).

9. Ley de Administración

Sun le informa que Cualquier acción relacionada con el presente acuerdo se regirá por la ley de California de EE.UU. y el control de la ley federal. Ninguna de las reglas de ley de cualquier jurisdicción será aplicada.

10. Divisibilidad

Si alguna disposición de este acuerdo no puede entrar en vigor, el presente acuerdo permanecerá en vigor con la disposición omitida, a menos que la omisión afectara a la intención de las partes, en cuyo caso el presente acuerdo cesará inmediatamente.

11. Integración

Este acuerdo es el acuerdo de compromiso entre usted y Sun en relación al tema del mismo. El mismo reemplaza todos los contratos anteriores y comunicaciones orales o escritas, propuestas, declaraciones y garantías y prevalece sobre cualquier conflicto o términos adicionales de cualquier cita, orden, reconocimiento, o de otro tipo de comunicación entre las partes relacionadas con la materia durante el periodo de vigencia de este acuerdo. Ninguna modificación del presente acuerdo será vinculante, excepto por un escrito firmado por un representante autorizado de cada parte.

Términos de licencia suplementarios

Estos términos de licencia suplementarios ("Términos Suplementarios") añaden o modifican términos del Acuerdo de Licencia de Código Binario (Colectivamente, el "Acuerdo"). Los términos no definidos en estos Términos Suplementarios tendrán el mismo significado que se le asigna a ellos en el propio acuerdo. Estos Términos Suplementarios sustituirán cualquier término inconsistente o conflictivo en el acuerdo, o en cualquier licencia contenida en este Software.

1. Uso interno del Software y concesión de licencia de desarrollo.

Sujeto a los términos y condiciones de este acuerdo pero no limitado a la Sección 3 (Restricciones de la Tecnología Java) de estos Términos Suplementarios, Sun le concede de manera no exclusiva e intransferible, licencia limitada para internamente reproducir y usar la forma binaria del Software, completa y sin modificar, para el único propósito de diseñar, desarrollar y probar tus applets de Java y aplicaciones ("Programas").

2. Licencia para distribuir Software

Además de la licencia concedida en la Sección 1 (Uso interno del Software y concesión de licencia de desarrollo) de estos Términos Suplementarios, sujeto a los términos y condiciones de este acuerdo, incluyendo pero no limitando a la Sección 3 (Restricciones de la tecnología Java) de estos Términos Suplementarios, Sun le concede licencia

limitada no exclusiva y intransferible para reproducir y distribuir el Software solo en forma de código binario, siempre y cuando:

i. Se distribuya el Software completo y sin modificar, con la posibilidad de omitir los archivos especialmente identificados como "opcional" en el archivo "README" del Software, los cuales incluyen ejemplos, documentos y archivos binarios, o que son eliminables mediante el uso de la herramienta de personalización de Software suministrada, sólo como parte de y para único propósito de ejecutar el Programa en el que se ejecuta el Software;

ii. No se distribuya Software adicional con la intención de remplazar cualquier componente del Software;

iii. No se elimine o altere cualquier leyenda o aviso de propiedad de contenidos del Software;

iv. Solo se distribuya el Software sujeto a los acuerdos de licencia que protege los intereses de Sun con los términos contenidos en ese acuerdo;

v. Se comprometa a defender e indemnizar a Sun y sus licenciatarios de y contra cualquier daño, coste, obligación, cantidades o gastos de liquidación, incluyendo los honorarios de abogados, incurridos en relación a cualquier reclamación, demanda o acción de cualquier tercera parte que surja o resulte del uso de la distribución de cualquiera de los programas del software.

3. Restricciones de la tecnología Java.

No puede modificar la Interfaz de la Plataforma Java ("JPI", identificados como clases que figuran en el paquete "java" o en cualquier sub-paquete del paquete "java"), creando clases adicionales dentro de JPI o de otro modo causando la adicción o modificación de las clases en el JPI. Aplicable en el caso de crear una clase adicional y en las APIs asociadas, las cuales:

i. Amplían la funcionalidad de la plataforma Java, y

ii. si está expuesta a terceros desarrolladores de software con el propósito de desarrollar software adicional que invoque dicha API adicional, se debe publicar de inmediato en términos generales una especificación precisa para dicha API p para el uso gratuito de todos los desarrolladores. No se puede crear o autorizar a sus licenciatarios para crear clases adicionales, interfaces, paquetes o sub-paquetes que estén de alguna forma identificados como "java", "javax", "sun" o convenios similares especificados por Sun en cualquier archivo de clases de convención de nomenclatura de designación.

4. Disponibilidad de Java Runtime

Sun le facilita la consulta de la versión adecuada de licencia de código binario de Java Runtime Environment (Actualmente situado en http://www.java.sun.com/jdk/index.html)para la disponibilidad del código de ejecución el cual puede distribuirse con applets de Java y aplicaciones.

5. Marcas y Logos.

Sun le obliga a aceptar y reconocer que entre que Sun es propietaria de las marcas SUN, SOLARIS, JAVA, JINI, FORTE, STAROFFICE, STARPORTAL e iPLANET y todas las marcas relacionadas, servicios de marca, logos y otras designaciones de marca de SUN, SOLARIS, JAVA, JINI, FORTE, STAROFFICE, STARPORTAL e iPLANET ("Sun Marks"), y le compromete a cumplir con la Sun Trademark y los Requisitos de Uso, actualmente situados en http://www.sun.com/policies/trademarks . Sun le informa que cualquier cambio que haga en las marcas de Sun sólo influye a los beneficios de Sun.

6. Código fuente.

El Software puede contener código fuente que se proporciona únicamente con fines de referencia de conformidad con los términos del presente acuerdo. El código fuente no podrá redistribuirse a menos que se disponga expresamente en el presente acuerdo.

7. Terminación por infracción.

Cualquiera de las partes integrantes de este acuerdo puede terminarlo de forma inmediata en caso de que cualquiera de las partes considere que el software puede ser objeto de una alegación de infracción de cualquier derecho de propiedad intelectual.

Anexo 3

Manual de usuario de las aplicaciones desarrolladas

Introducción

Este manual le permitirá aprender a utilizar las funcionalidades del proyecto (Tratamiento inteligente de imágenes para la manipulación de un mundo virtual).

El programa se centra en el tratamiento de imágenes a tiempo real, es decir, imágenes capturadas por una WebCam. El tratamiento consistirá en la detección de un objeto rojo y su movimiento.

Requerimientos básicos

Para PC

Componentes	Qué necesitas (Mínimo)	Qué necesitas (Recomendado)
Procesador	PC a 233MHz o superior (Intel Pentium II o superior)	PC a 500MHz o superior
RAM	128MB	512MB o superior
Sistema Operativo	2000/ME/XP	Windows XP
Tarjeta de vídeo	16-bit (alta resolución) o superior	24-bit (color verdadero) o superior
Resolución de la pantalla	800x600	1024x768 o superior

Para ordenadores Mac

Componentes	Qué necesitas (Mínimo)	Qué necesitas (Recomendado)
Procesador	Procesador PowerPC G3 a 400MHZz	Procesador PowerPC G4 500MHz o superior
RAM	128MB	256MB o superior
Sistema operativo	Mac OS 10.3.9	Mac OS 10.4 o superior
Tarjeta de vídeo	memoria de vídeo de 16MB	memoria de vídeo de 64MB o superior
Resolución de pantalla	800x600	1024x768 o superior
Reproductor Multimedia	QuickTime Versión 7.0	QuickTime Versión 7.0.3 o superior

Se requiere una WebCam que soporte la resolución de 640x480.

Lo primero es tener la última versión del JRE y del JDK para instalar el Java Runtime Environment.

Ya con eso instalado ahora es turno del Java Media Framework (en español, Entorno de Trabajo Multimedia de Java) es una extensión de Java que permite la programación de tareas multimedia en este lenguaje de programación.

Se descarga desde:
http://java.sun.com/javase/technologies/desktop/media/jmf/2.1.1/download.html

JMF no se incluye en la JDK, ni en la JRE, sino que debe conseguirse como un paquete externo.

Aplicación

Es importante para utilizar el programa tener un objeto rojo, que no brille ni refleje mucho la luz. Que no sea muy pequeño de manera que en la escena pueda existir otro objeto rojo, que se considere más importante, y se ignore el *objeto principal*. Ni tan grande que ralentice el tratamiento de las imágenes, ya que esto conlleva a tratar una proporción muy grande de píxeles rojos, con lo que afecta la velocidad de respuesta de la aplicación.

El objeto ideal utilizado por quienes hemos hecho la aplicación, después de haber realizado innumerables pruebas, es una nariz de payaso, que colocamos sobre un bolígrafo de otro color.

Para trabajar con el programa inicialmente tenemos la opción Capture (captura de la cámara Web) o salir de la aplicación.

Figura 1: Pantalla inicial

Una vez elegida la opción de Capture se nos presenta una pantalla para configurar las opciones de la captura de la cámara Web.

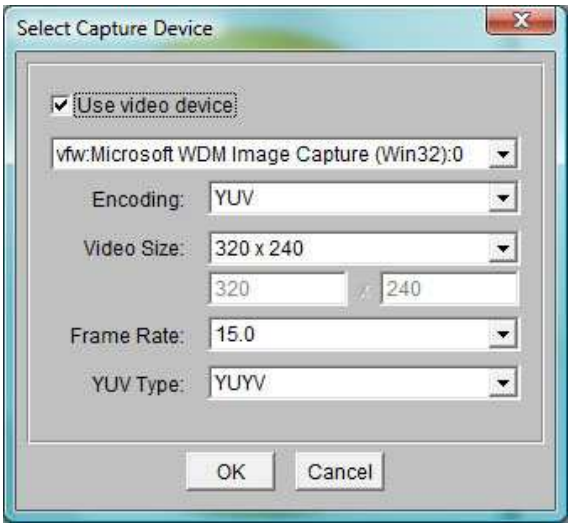


Figura 2: Formulario de selección de opciones

La opción Use video device (uso del dispositivo de video) debe estar seleccionada.

La codificación YUV (Encoding: YUV) es la única utilizada. YUV es un método que define a una señal de video que separa los componentes de luminosidad (Y) y color (UV). Individualmente, las letras YUV significan Intensity, Hue, y value. (Intensidad, matíz y valor). El proceso de codificación YUV toma ventajas de este fenómeno y provee un ancho de banda mayor para la información de luminancia que para la de crominancia.

En la siguiente imagen también, se ve como podría variar el tamaño de video.

Las opciones del tamaño del video así como la frecuencia de refresco influyen directamente en la velocidad de ejecución de la aplicación.

En este caso es importante tener en cuenta que al aumentar el tamaño de la imagen del video, aumenta el número de píxeles a tratar posteriormente por cada imagen, lo que determina la respuesta de la aplicación.

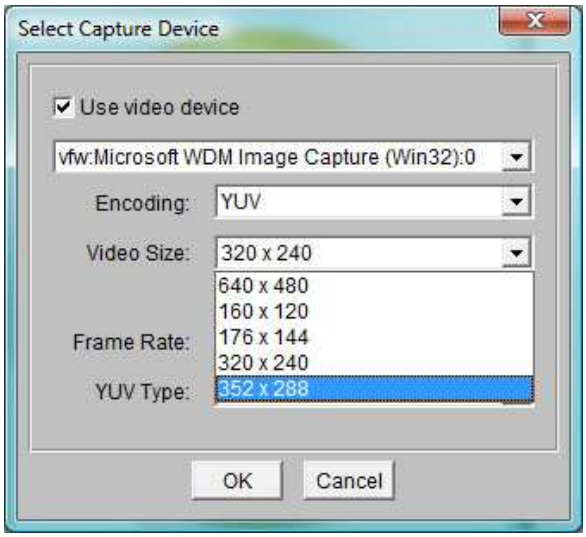


Figura 3: Selección del tamaño del video

Finalmente se ejecuta el programa. Por un lado esta el formulario donde se muestra el comportamiento del objeto tratado y por otro lado se ve la captura de la WebCam en ese momento, en tiempo real.

Este proyecto consta de varias partes o aplicaciones diferentes, pero en todas ellas se cumplen los pasos antes mencionados.

Aplicación I

En la primera aplicación del proyecto o parte uno, nos encontramos que una vez que se ejecuta el programa, a parte de la pantalla que muestra el video en el que nos vemos en ese momento, nos tenemos un formulario donde podemos dibujar la trayectoria de un objeto rojo.

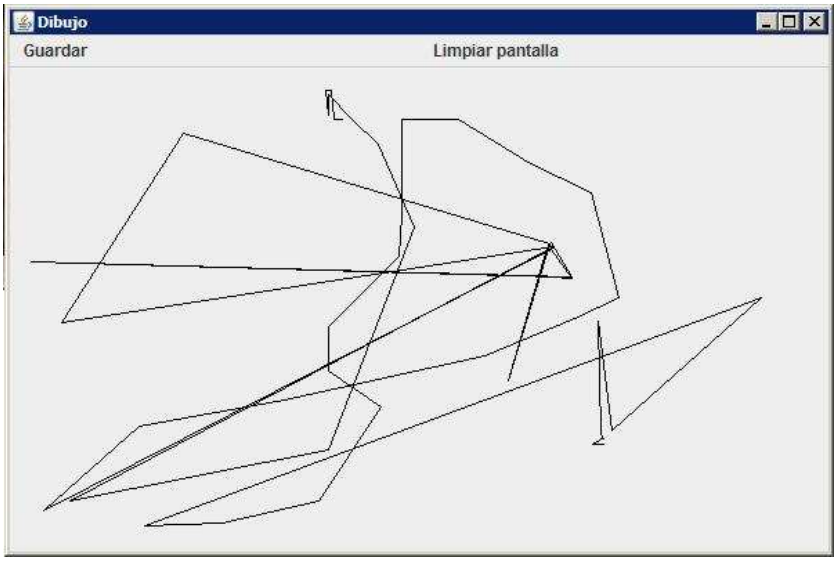


Figura 4: Proyecto 1, representación de la trayectoria

El formulario cuenta con dos opciones, guardar una imagen del mismo o limpiar la pantalla para volver a empezar.

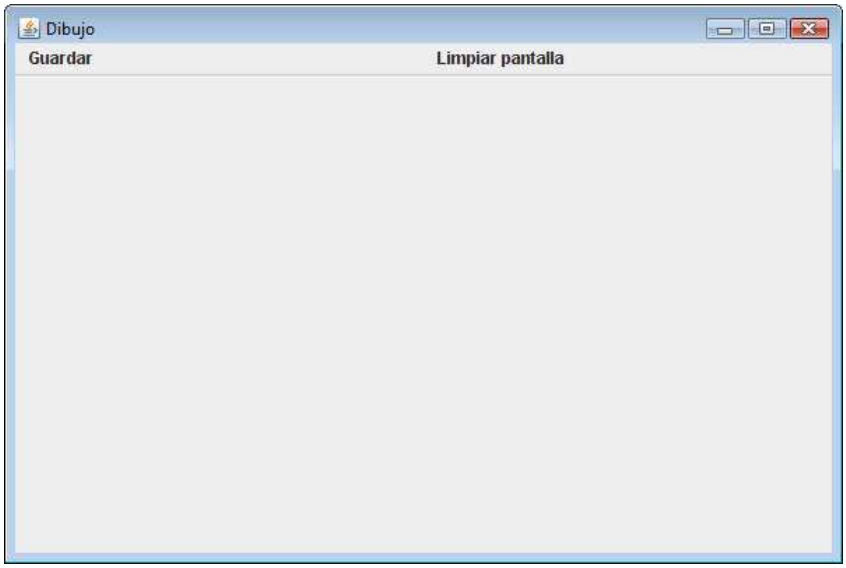


Figura 5: Muestra opción "Limpiar pantalla" del formulario gráfico

Del mismo modo que se puede representar el movimiento del objeto, podemos guardar la traza como una imagen en formato jpg.

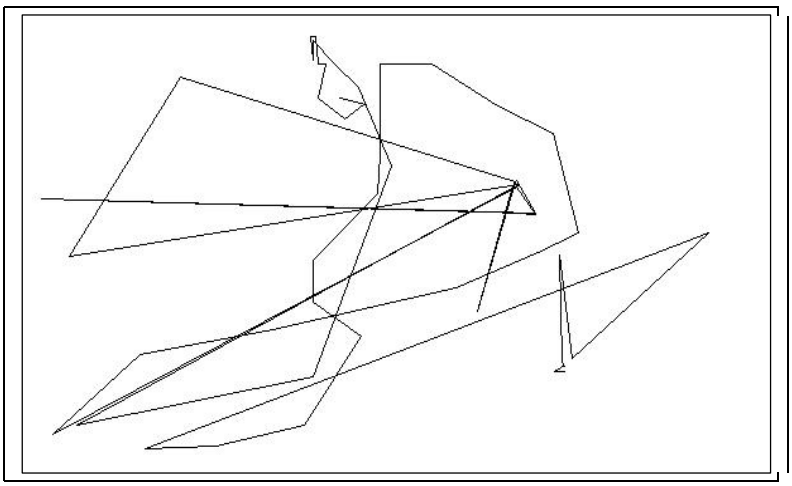


Figura 6: Muestra opción: "guardar".Imagen guardada en .jpg

Aplicación II

La fase dos del proyecto, es una aplicación del caso anterior, una vez detectada la trayectoria de un objeto, se intenta simular un escritorio de manera virtual, donde hay carpetas que pueden ser seleccionadas mediante un clic del ratón y desplazadas a otra posición dentro del escritorio.

Si lo que se desea es ejecutar esta fase del proyecto, la pantalla inicial será la que se muestra a continuación.

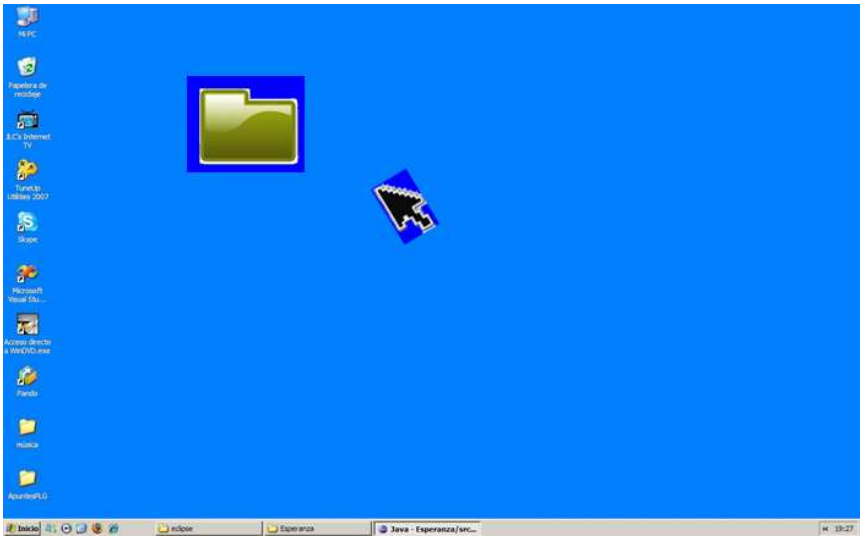


Figura 7: Escritorio virtual

El objetivo es seleccionar la carpeta verde que se muestra en la imagen con el cursor. Este cursor es en si el objeto rojo captado por la WebCam.

Para seleccionar la carpeta hay que hacer un clic sobre ella, o lo que es lo mismo, acercar el objeto rojo apuntando sobre la carpeta, en un periodo corto de tiempo.
Cuando se hace un clic sobre la carpeta, la imagen de esta cambia, con lo que podemos estar seguros que el clic se ha llevado a cabo.

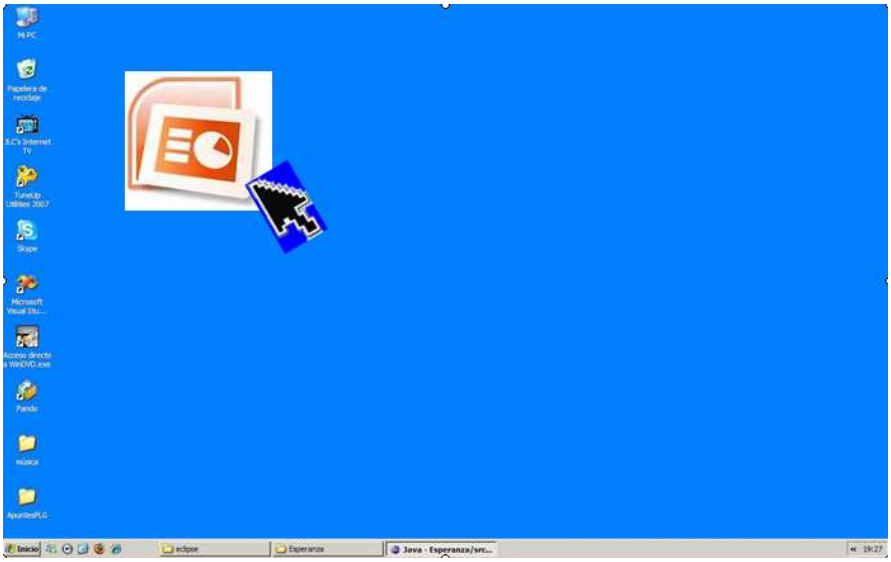


Figura 8: Selección de la carpeta

Una vez que se ha hecho clic sobre la carpeta, volvemos a hacer un clic sobre la parte del escritorio (formulario) donde queremos que se desplace la carpeta.

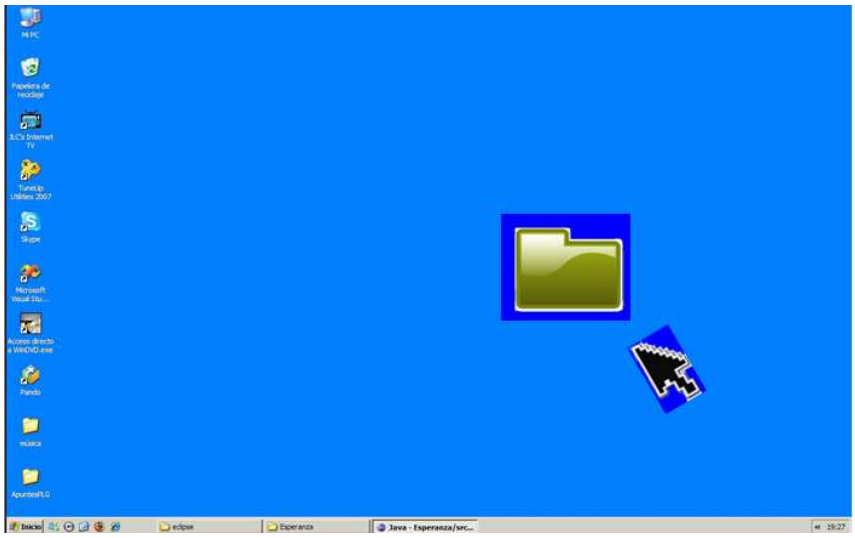


Figura 9: Desplazamiento en el escritorio

Y vuelta a empezar.

Aplicación III

Se basa en la simulación muy sencilla del teclado de un teléfono, en donde para realizar una llamada seleccionamos la secuencia de números deseados, pulsamos sobre la tecla llamar, para finalizar la llamada seleccionamos colgar.

A continuación se muestra la pantalla inicial con la representación del teclado.

Hay que recordar que la ejecución del programa se realiza de la misma forma que en los casos anteriores, es necesario utilizar un objeto rojo, que pueda ser detectado por la WebCam.

Figura 9: Teclado convencional de un teléfono

Cuando las teclas son seleccionadas, la apariencia de las mismas cambia, de manera que podemos ser conscientes que el número ha sido marcado. Del mismo modo, en la parte superior del formulario se muestra la secuencia de números que se lleva hasta el momento.

Figura 10: Acción marcar

Cuando se ha terminado de marcar el número de teléfono que se desea, para llamar hay que hacer un clic sobre la tecla ver, siguiendo el modelo de los móviles convencionales.

Figura 11: Acción llamar

Para finalizar la llamada basta con pulsar la tecla roja (colgar).

Figura 12: Acción colgar.

Parte 5. BIBLIOGRAFÍA

[1]. Inteligencia Artificial E Ingeniería Del Conocimiento. Pajares Martinsanz, Gonzalo. Ra-Ma - 2005

[2]. Tratamiento inteligente de imágenes para la manipulación de un mundo virtual. Kalbakdij, Silvia; Lebrero, Pablo;Sanchez,Salvador; Garmendia, Luis. Actas de las Jornadas internacionales de Didáctica de las Matemáticas en Ingeniería. E.T.S.I. Caminos, UPM. Páginas 305-311. 15-16 de junio de 2009.

[3]. http://java.sun.com/javase/technologies/desktop/media/jmf/

[4]. http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=eclipseUml2

[5]. http://atc.umh.es/gatcom/Ficheros/Articulos/Jenui2k.pdf